ESTE EXEMPLAR CORRESPONDE A REDAÇÃO FINAL DA TESE DEFENDIDA POR Augusto Seganfredo Mainardi E APROVADA PELA COMISSÃO JULGADORA EM 16 / 07 / 2010

ORIENTADOR

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Augusto Seganfredo Mainardi

Simulação de Dispositivos Robóticos Móveis com ênfase no Planejamento de Trajetórias para Navegação

Campinas, 2010.

70/2010

Augusto Seganfredo Mainardi

Simulação de Dispositivos Robóticos Móveis com ênfase no Planejamento de Trajetórias para Navegação

Dissertação apresentada ao Curso de Mestrado da Faculdade de Engenharia Mecânica da Universidade Estadual de Campinas, como requisito para a obtenção do título de Mestre em Engenharia Mecânica.

Área de Concentração: Mecânica dos Sólidos e Projeto Mecânico

Orientador: Prof. Dr. João Maurício Rosário

Campinas

2010

CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Mainardi, Augusto Segansfredo
M284s Simulação de dispositivos robóticos móveis com ênfase no planejamento de trajetórias para navegação / Augusto Seganfredo Mainardi. --Campinas, SP: [s.n.], 2010.

Orientador: João Maurício Rosário.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Robôs móveis. 2. Navegação de robôs móveis. 3. Simulação. I. Rosário, João Maurício. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

Título em Inglês: Mobile robotic devices simulation with emphasis in trajectory planning for navigation
Palavras-chave em Inglês: Mobile robots, Navigation of mobile robots, Simulation
Área de concentração: Mecânica dos Sólidos e Projeto Mecânico
Titulação: Mestre em Engenharia Mecânica
Banca examinadora: Jocarly Patrocinio de Souza, Ely Carneiro de Paiva
Data da defesa: 16/07/2010
Programa de Pós-Graduação: Engenharia Mecânica

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETOS MECÂNICOS

DISSERTAÇÃO DE MESTRADO ACADEMICO

Simulação de Dispositivos Robóticos Móveis com ênfase no Planejamento de Trajetórias para Navegação

Autor: Augusto Seganfredo Mainardi
Orientador: Prof. Dr. João Maurício Rosário

A Banca Examinadora composta pelos membros abaixo aprovou esta Dissertação:

Prof. Dr. João Maurício Rosário, Presidente
Universidade Estadual de Campinas

Prof. Dr. Jocarly Patrocinio de Souza
Universidade de Passo Fundo

Prof. Dr. Ely Carneiro de Paiva
Universidade Estadual de Campinas

Campinas, 16 de julho de 2010.

Dedicatória:

Dedico este trabalho à minha família e aos colegas que incentivaram e colaboraram com este trabalho e a todos que de alguma maneira poderão utilizá-lo e retirar dele algum benefício.

Agradecimentos

Este trabalho conta com a colaboração de várias pessoas que em diversos aspectos, e sem estas pessoas o trabalho não se viabilizaria, portanto, eu gostaria de agradecer:

- Aos meus familiares, pelo incentivo em todos os momentos;
- Ao meu orientador, pelo apoio, tempo e conhecimento transmitido;
- A todos os colegas do Laboratório de Automação Integrada e Robótica (LAIR), que ajudaram de forma direta e indireta na conclusão deste trabalho, e um agradecimento especial ao Álvaro, ao Luciano e ao Marcos.
- A todos os funcionários da Faculdade de Engenharia Mecânica da UNICAMP, em especial ao Miro, à Juliana, à Denise e à Silvana;
- A todos os meus amigos que participaram da minha estadia em Campinas, em especial ao Miguel, Cleiton, Edson, Fabrício, Eduardo, Felipe, Bruno, Pedro e Guilhermes.
- E à CAPES, pelo financiamento durante a realização da dissertação de mestrado.

“O homem realmente culto
não se envergonha de fazer perguntas
também aos menos instruídos.”
(Lao Tsé)

Resumo

MAINARDI, Augusto Seganfredo, *Simulação de dispositivos robóticos móveis com ênfase no planejamento de trajetórias para navegação*; Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2010. 221 p. Dissertação (Mestrado).

Neste trabalho é proposto um sistema de navegação autônomo para dispositivos robóticos móveis capaz de operar e se adaptar a diferentes ambientes e condições, contribuindo para o desenvolvimento de uma navegação robusta e confiável. O sistema é baseado na arquitetura híbrida AuRA, assim, foi separado em quatro componentes: percepção do ambiente, localização e mapeamento, planejamento de movimento e execução da trajetória. A percepção do ambiente é o componente responsável em converter as leituras dos sensores em informações sobre o ambiente. Considerando os sensores usadas da plataforma robótica móvel ASURO, este componente baseia-se na informações obtidas através da odometria e dos sensores seguidores de linha, informando ao sistema a distância percorrida e a posição do robô em relação a pista a ser seguida. O mapeamento do sistema baseou-se em mapas topológicos devido ao baixo custo computacional necessário e à semelhança com a maneira humana de localizar-se, utilizando a odometria como sistema de localização do robô e sensores seguidores de linha para determinação de seu posicionamento. O planejamento de movimentos foi dividido em duas fases. No planejamento de caminho utilizou-se o algoritmo de Dijkstra para determinar por quais nós ele deve passar para atingir seu objetivo; e para o planejamento de trajetória utilizou-se uma abordagem baseada no caminho de Dubins. A execução da trajetória baseou-se no método de Motor-Schemas, onde as respostas dos atuadores são determinadas pela soma vetorial dos vetores resultantes de cada comportamento. Foram estudadas duas formas de comportamento: o de seguir o objetivo que utiliza o planejamento de movimento para determinar as velocidades dos atuadores; e o de seguir uma linha, que utiliza a percepção do ambiente para determinar as velocidades dos atuadores. As implementações experimentais foram realizadas a partir do ambiente de simulação DD&GP desenvolvido para o ambiente MATLAB-Simulink®, que permitiu a avaliação do sistema a partir de duas aplicações (transporte e inspeção) efetuada em três ambientes diferentes (fábrica, escritório e sistema de tubulação). Além disso, utilizou-se a plataforma robótica móvel ASURO para verificar a percepção do ambiente e validar os resultados encontrados nas simulações. Os resultados obtidos nas implementações experimentais foram satisfatórios e mostram que o sistema apresentado é promissor.

Palavras Chave: Robôs Móveis, Navegação de Robôs Moveis e Simulação.

Abstract

MAINARDI, Augusto Seganfredo, *Mobile robotic devices simulation with emphasis in trajectory planning for navigation*; Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2010. 221 p. Dissertação (Mestrado).

In this work is proposed an autonomous navigation system for mobile robotic devices able to operate and adapt to different environments and conditions, contributing to the development of a robust and reliable navigation system. The system is based on hybrid architecture AuRA, thus, it was separated into four components: Perceptions of the environment, Localization and Mapping, Motion planning and Trajectory execution. The perception of the environment is the component responsible for converting the readings in sensors in environmental information. Considering the sensors used in mobile robotics platform ASURO, this component is based on information obtained from odometry and line following sensors, informing the system the distance traveled and the robot's position in relation to the track to be followed. The mapping of the system is based on topological maps due low computational cost required and its resemblance to the human way of locating themselves and the use of little computer memory, using the odometry as robot's localization system and line following sensors to determine their placement. The Motion planning was divided into two phases. In path planning was used Dijkstra's algorithm to determine for which node the robot must pass to achieve your goal; and for trajectory planning was used an approach based on Dubins path. The trajectory execution is based on the method of motor-schemas, where the responses of the actuators are determined by the vector sum of the resulting vectors from each behavior. Were studied two forms of behaviors: follow the goal, which uses the motion planning to determine the velocity of actuators; and follow a line, which uses the perception of the environment to determine the velocity of actuators. The experimental implementations were realized from the simulation environment DD&GP developed for the MATLAB-Simulink ®, which allowed the evaluation of the system after two applications (transport and inspection) performed in three different environments (factory, office and piping system). In addition, was used the platform for mobile robotics ASURO to verify the perception of the environment and validate the results found in the simulations. The results obtained in experimental implementations were satisfactory and showed that the system presented is promising.

Key Words: Mobile Robots, Mobile Robot Navigation and Simulation.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

Letras Latinas

c - velocidade do som	[m/s]
d - distância entre dois corpos	[m]
E – trajetos	
G – grafo	
K – ganho	
l – distância entre as rodas	[m]
M – leitura dos fotosensores	[bits]
n – tamanho do vetor	
N – nós	
p – posição	
P – peso do comportamento	
r – raio da roda	[m]
R – distância entre o centro de massa do robô e o CCI	[m]
S – distância entre a linha central do robô e a linha central da pista	[m]
t – tempo	[s]
v – nós	
V – velocidade linear	[m/s]

.....

Letras Gregas

φ – facilidade do trajeto	
σ – fator de segurança	
ξ – matriz de postura do robô	
ν – parâmetro de direção	
α – ângulo de inclinação do objetivo	[rad]
γ – ângulo de curvatura	[rad]

θ – orientação angular inicial [rad]

μ – velocidade do deslocamento lateral [m/s]

τ – distância entre o ponto de auxílio e o nó [m]

ψ – pesos dos trajetos

ω – velocidade angular [rad/s]

.....

Superescritos

® - marca registrada;

TM – marca registrada comercial;

.....

Subscritos

d – direita;

D – direita;

e - esquerda;

E - esquerda;

.....

Abreviações

DesR – quantidade de deslocamento do robô;

largp – largura da linha;

ODOM – leitura dos odômetros.

.....

Siglas

3T – *3 Tiers* (3 Camadas);

A* – AStar (Algoritmo de Procura)

ASURO – *Another Small and Unique Robot from Oberpfaffenhofen*;

AuRA - *Autonomous Robot Architecture* (Arquitetura de Robôs Autônomos);

BFS - Breadth-First Search (Busca em Largura-Primeiro ou Busca em Amplitude)

CCI - Centro de Curvatura Instantâneo

CDRT – Cálculo da Distância entre o Robô e a Trajetória desejada;

D* - *Dynamic A** (A* Dinâmico);

DAMN – *Distributed Architecture for Mobile Navigation* (Arquitetura Distribuída para a Navegação Móvel);

DC – *Direct Current* (Corrente Contínua);

DD&GP – *Differential Drive and Global Positinong* (Direção Diferencial e Posicionamento Global);

DD&P – *Dual Dynamics and Planning* (Dual Dinâmica e Planejamento);

DLR – *Deutsches Zentrum für Luft- und Raumfahrt* (Centro Aeroespacial Alemão);

DP – *Distinctives Places* (Pontos Distintos);

DSP – *Digital Signal Processor* (Processador de Sinais Digitais);

GPS – *Global Positioning System* (Sistema de Posicionamento Global);

GUI – *Graphical User Interface* (Interface Gráfica do Usuário)

LCD – *Liquid Crystal Display* (Tela de Cristal Líquido);

LEMV – *Long Endurance Multi-Intelligent Vehicle* (Veículos Multi-Inteligentes para viagens de Longa Duração);

MIT – *Massachusetts Institute of Technology* (Instituto de Tecnologia de Massachusetts);

NASA – *National Aeronautics and Space Administration* (Administração Nacional do Espaço e da Aeronáutica);

NHC - *Nested Hierarchical Controller* (controlador Hierárquico Aninhado);

NIST – *National Institute of Standards and Technology* (Instituto Nacional de Padrões e Tecnologia);

PID – Proporcional –Integral –Diferencial;

PRS – *Procedural Reasoning System* (Sistema de Raciocínio Procedural)

RCS - *Realtime Control System* (Sistema de Controle em tempo Real);

RUR - *Rossum's Universal Robots* (Robôs Universais de Rossum);

SFX – *Sensor Fusion Effects* (Efeitos de Fusão Sensorial);

SLAM – Simultaneos Localization and Mapping (Localização e Mapeamento Simultâneos);

SRI - *Stanford Research Institute* (Instituto de Pesquisa de Stanford);

SSS – *"Servo, Subsumption, Symbolic" systems* (sistemas "Servo, Subsumption, Simbólico");

STRIPS – *Stanford Research Institute Problem* Solver (Resolvedor de Problemas do Instituto de Pesquisas de Stanford);

USB – *Universal Serial Bus* (Barramento Serial Universal).

.....

SUMÁRIO

CAPÍTULO 1

Introdução

Dispositivos robóticos vêm sendo utilizados em grande escala na indústria, onde os robôs manipuladores operam na soldagem, pintura e montagem de materiais, devido sua velocidade e precisão. Contudo, a falta de mobilidade dos robôs manipuladores restringe sua utilização, mostrando a necessidade do surgimento de outra classe de robôs, os robôs móveis (SIEGWART e NOURBAKHSH, 2004).

Enquanto os robôs manipuladores são empregados para deslocar objetos definidos numa área de trabalho restrita através de movimentos conhecidos, os robôs móveis devem se deslocar (carregando ou não um objeto) numa área de trabalho ilimitada e desconhecida, tendo que lidar com incertezas, como diferenças no solo e derrapagens das rodas, tornando a seqüência de movimentos desconhecida.

Assim, essa nova classe de robôs, os robôs móveis, devem ser constituídos por uma plataforma capaz de mover-se automaticamente num ambiente, percebendo e reagindo às mudanças no ambiente. O que, devido a capacidade de interagir com diversos ambientes, torna o estudo em robótica móvel de grande importância não apenas para a indústria, como também para as aplicações de tecnologias no dia-a-dia.

Segundo Nehmzow (2000), um robô móvel autônomo tem a capacidade de movimentar-se, adaptar-se, e perceber o ambiente em que estiver inserido através de seus sensores de modo a aprender e construir representações internas do seu ambiente que possam ser usadas no seu processo de tomada de decisão.

Aprofundando o conceito de robô móvel, Marchi (PIERI, 2002) define o robô móvel como um dispositivo mecânico montado sobre uma base não fixa que age sob o

controle de um sistema computacional, equipado com sensores e atuadores que o permitem interagir com o ambiente.

Finalmente, Bianchi (2007) propôs que um robô móvel pode ser decomposto em três elementos: um mecanismo de locomoção, um ou mais computadores para o controle, e uma coleção de sensores com os quais o robô obtém informação do ambiente.

Portanto, podemos definir um robô móvel como uma plataforma móvel dotada de uma coleção de sensores, um sistema computacional e um conjunto de atuadores, com os quais é apto a compor uma representação interna do ambiente em que está inserido (possuindo ou não uma representação inicial do ambiente), utilizando-a para o seu processo de tomada de decisão, e assim, interagir com o ambiente coerentemente.

Para isso, o estudo dos robôs móveis torna-se uma área de pesquisa predominantemente interdisciplinar envolvendo as seguintes áreas de conhecimento (DUDEK e JENKIN, 2000):

- **Engenharia Mecânica:** Projeto mecânico dos veículos e das partes envolvidas, particularmente os mecanismos de locomoção.
- **Engenharia Elétrica:** Projeto e integração dos sistemas elétricos e eletrônicos, dos sensores e da comunicação.
- **Ciência da Computação:** Desenvolvimento de programas, simuladores, algoritmos para modelagem, sensoriamento e planejamento.
- **Psicologia Cognitiva, Percepção e Neurociência:** Entender como sistemas biológicos resolvem problemas similares aos desejados.

1.1 Motivação do Trabalho

O desenvolvimento de dispositivos robóticos móveis vem sendo pesquisados atualmente, sendo apresentados diversas abordagens para a implementação de um sistema de navegação. Porém, os sistemas desenvolvidos geralmente operam em ambientes

controlados, e os sistemas desenvolvidos para serem utilizados no mundo real são limitados e não apresentam um comportamento autônomo (HEINEN, 2002). Deste modo, apesar do avanço nesta área ter aprimorado a navegação robótica, aumentando massivamente o número de aplicações, a "realidade dos Jetsons" está muito distante da realidade atual, já que as pesquisas até o momento não têm apresentado uma navegação robusta e confiável capaz de atuar num ambiente complexo e dinâmico como o mundo real. Assim, ainda é necessário intensificar a pesquisa nessa área, de modo que num futuro próximo possamos obter uma navegação apropriada a utilização em fábricas, escritórios e em nossas casas. Portanto, a principal motivação desta dissertação é contribuir para o desenvolvimento de uma navegação robusta e confiável.

1.2 Objetivos

Este trabalho tem como objetivo principal desenvolver e simular um sistema de navegação autônoma para dispositivos robóticos móveis, capaz de operar e de se adaptar a diferentes ambientes e condições, com ênfase no planejamento de trajetória.

Objetivos específicos

Os principais objetivos deste trabalho de pesquisa são os seguintes:

- Analisar diferentes arquiteturas de controle de robôs móveis, selecionando a mais adequada à proposta;
- Estudar as técnicas utilizadas para a navegação robótica, avaliando os benefícios e limitações de cada técnica;
- Propor um conjunto de técnicas para a formação de um sistema de navegação;
- Simular o planejamento de movimento em três ambiente (ambiente fabril, escritório administrativo e uma rede de tubulação industrial) considerando duas abordagens, inspeção e transporte, de forma a validar as abordagens escolhidas.

1.3 Justificativa do Trabalho

A capacidade dos robôs móveis de interagir em diversos ambientes torna-os de grande utilidade no dia-a-dia humano, pois uma navegação autônoma é atrativa do ponto de vista industrial, devido à precisão, coordenação e velocidade que estes dispositivos proporcionam, e do ponto de vista individual, de modo a simplificar e facilitar o dia-a-dia em casa, desde a realização de tarefas domésticas como no auxílio de pessoas com necessidade especiais. Porém, os métodos de navegação robótica existentes ainda não são robustos e confiáveis, sendo necessário um maior desenvolvimento, contudo, a pesquisa nesta área possui um grande limitador, que é o alto custo da plataforma robótica e da infra-estrutura necessária. Dessa forma, a utilização de simulações e uma plataforma de baixo custo superaram essa restrição, e demonstra a possibilidade aumentar o nível da navegação com gastos diminutos.

1.4 Organização do trabalho

O presente documento encontra-se organizado a partir de metodologia que permita atingir os objetivos inicialmente estabelecidos. Com o propósito de estudar e analisar alguns conceitos envolvidos no desenvolvimento de um robô móvel no capítulo 2 desta dissertação de mestrado é realizado um trabalho de revisão bibliográfica aprofundada sobre os elementos mecânicos e elétricos de uma plataforma robótica móvel. Como complemento a este capítulo são apresentados, sob a forma de quatro anexos, alguns tópicos referentes aos sistemas robóticos móveis: Arquitetura de Controle, Mapeamento e Localização, Planejamento de Movimentos e Sistemas de Navegação.

No capítulo 3 desse trabalho é apresentado o sistema de navegação proposto, onde são abordados e exemplificados as técnicas discutidos no capitulo 2 e nos anexos deste documento.

No capítulo 4 são apresentados os principais resultados obtidos, destacando-se a implementação em software de modelagem dinâmica e controle de sistemas robóticos

móveis desenvolvidos em MATLAB-Simulink®, e os principais resultados experimentais obtidos a partir da plataforma robótica móvel ASURO™.

No capítulo 5 são apresentadas as principais conclusões deste trabalho de pesquisa, sendo também discutidos e propostos alguns trabalhos futuros de pesquisa nesta área.

CAPÍTULO 2

Revisão Bibliográfica

Um dispositivo robótico móvel utiliza as informações pré-estabelecidas e as informações dos sensores da plataforma móvel para realizar o controle deste sistema mecânico. Desta forma, torna-se imprescindível para o correto controle de posicionamento do robô o conhecimento de seus sensores e do modelo cinemático da plataforma robótica em estudo.

Neste capítulo é realizado um trabalho de revisão bibliográfica aprofundado na área de Robótica Móvel, considerando a classificação de dispositivos robóticos móveis e uma descrição das principais plataformas robóticas disponíveis no mercado e suas características em relação à modelagem, sensoriamento e principais utilizações. No final deste capítulo é apresentada a plataforma robótica Móvel ASURO, utilizada como base de estudo experimental deste trabalho de pesquisa.

2.1 Classificação dos Dispositivos Robóticos Móveis

Considerando a possibilidade das diferentes configurações de robôs móveis, é inimaginável a multiplicidade de aplicações possíveis, permitindo assim que um dispositivo robótico móvel possa ser utilizado em diferentes tipos de ambientes, desde internos, como transporte, atendimentos, limpeza e entretenimento; até ambientes externos, como em uso espacial, militar, no deserto e agricultura (SICILIANO e KHATIB, 2008).

Atualmente são encontradas algumas aplicações, como em indústrias que utilizam plataformas robóticas para transporte, em casa onde existem robôs autônomos para limpeza (IROBOT, 2010) e para cortar a grama (HUSQVARNA, 2010), e, recentemente, outras pré-aplicações são vistas em exposições, como robôs para entretenimento e robôs

domésticos mais avançados (IRT, 2010; NOURBAKHSH, KUNZ e WILLEKE, 2003; ROBÓTICA-12C, 2010). A literatura atual subdivide os sistemas robóticos móveis em três grupos:

- **Robôs aéreos:** São dispositivos robóticos aptos a voar, utilizando geralmente hélices para se movimentar. Como exemplos de aplicações podemos destacar o robô aéreo MQ-9 Reaper (Figura 2.1a), que é utilizado pelo exército americano (PODER AÉREO, 2010); o modelo "Quadri France", fabricado pela empresa Taser France™ (Figura 2.1b), utilizado pela polícia francesa (TERRA TECNOLOGIA, 2008); o Long Endurance Multi-Intelligent Vehicle (LEMV) (Figura 2.1c), dirigível a ser utilizado pelo exército americano para uma cobertura radar e imagem em grandes aéreas (GIZMODO, 2010); e o primeiro micro-robô aéreo do mundo (Figura 2.1d), construído por um grupo de engenheiros liderados pelo prof. Mir Behrad Khamessee (TOP NEWS, 2009).

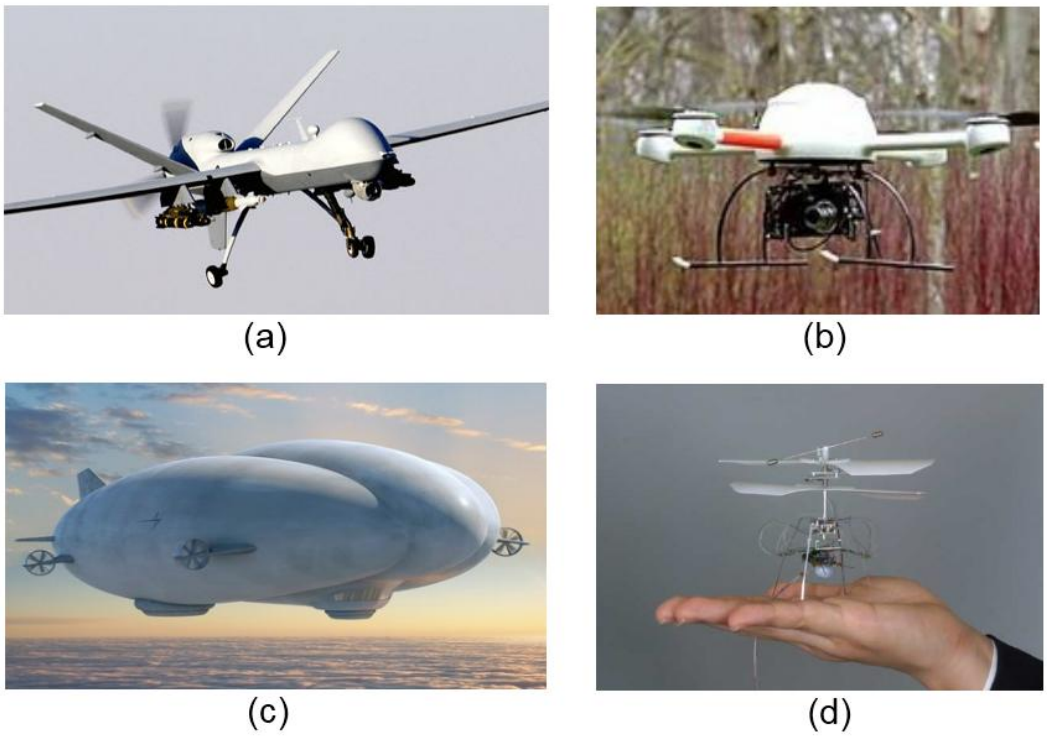


Figura 2.1 – Robôs aéreos. (a) MQ-9 Reaper (PODER AÉREO, 2010); (b) Quadri France (TERRA TECNOLOGIA, 2008); (c) LEMV (GIZMODO, 2010); (d) Micro robô aéreo (TOP NEWS, 2009).

- **Robôs aquáticos:** São dispositivos robóticos utilizados para se locomover sob ou sobre a água. A Figura 2.2a mostra o robô aquático AQUA construído na universidade McGill (CIM, 2010); e a Figura 2.2b mostra o robô Protector, desenvolvido em 2005 pela Marinha da República de Singapura, juntamente com seus navios *Endurance* para atuarem em missões de vigilância e reconhecimento no Norte do Golfo Pérsico (PROTECTOR, 2010).

Figura 2.2 – Robôs aquáticos. (a) AQUA (CIM, 2010); (b) PROTECTOR (PROTECTOR, 2010).

- **Robôs terrestres:** São dispositivos robóticos que utilizam o solo para se locomover, podendo ser divididos em quatro categorias (rodas, pernas, esteiras e rastejantes), de acordo como o atuador utilizado para a locomoção. A Figura 2.3 mostra cada uma das quatro categorias, onde a Figura 2.3a mostra o robô jBot, com rodas utilizado para navegação *off-road* (ANDERSON e HAMILTON, 2010); a Figura 2.3b mostra o BigDog, robô desenvolvido pela Boston Dynamics para se locomover em ambientes adversos (BIGDOG, 2010); a Figura 2.3c mostra o Dragon Runner, robô desenvolvido para o combate ao terrorismo (DEFENSE UPDATE, 2010); e a Figura 2.3d mostra o ACM-R3, robô desenvolvido pelo prof. Shigeo Hirose para auxiliar em resgates (THE TRIBUNE, 2010).

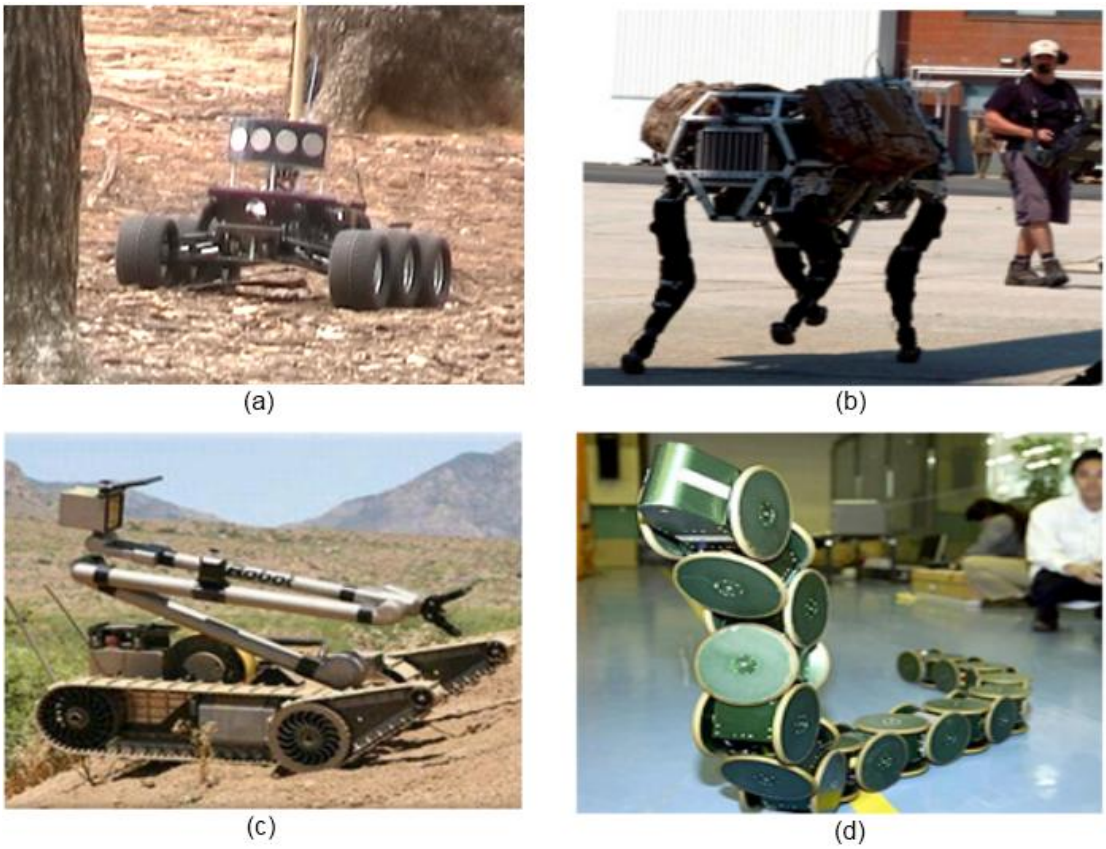


Figura 2.3 – Robôs terrestres. (a) jBot (ANDERSON e HAMILTON, 2010); (b) BigDog (BIGDOG, 2010); (c) Dragon Runner (DEFENSE UPDATE, 2010); (d) ACM-R3 (THE TRIBUNE, 2010).

2.2 Histórico

O primeiro robô conhecido foi construído em 1495, por Leonardo Da Vinci (ROSHEIM, 2006), mas o termo robô só foi introduzido pela primeira vez em 1921, quando o escritor checo Karel Čapek utilizou a palavra “robot” em sua peça "R.U.R” e cuja origem deriva da palavra checa “robota”, que significa "trabalho forçado”, entretanto, o criador do termo "robot" foi seu irmão Josef, outro respeitado escritor checo.

Em 1948, Grey Walter, da Universidade de Bristol, criou o primeiro robô autônomo eletrônico (VUKOBRATOVIC, 2007), e em 1956, logo após a segunda guerra mundial, dá-se inicio à pesquisa e desenvolvimento da inteligência artificial com o artigo "Computing

Machinery and Intelligence" do matemático inglês Alan Turing (CHARNIAK e McDERMOTT, 1985).

Em 1950 e 1983, Isaac Asimov publicou, os livros “Eu, Robô” e “Os robôs do amanhecer”, se tornando o escritor mais conhecido pelos roboticistas devido as Leis da Robótica introduzidas por ele nestes livros. As quatro Leis da Robótica enunciadas por Asimov, e amplamente aceitas pela comunidade (CLARKE, 1994) são as seguintes:

- 'Lei Zero': Um robô não pode fazer mal à humanidade e nem, por inação, permitir que ela sofra algum mal. Desse modo, o bem da humanidade é primordial ao dos indivíduos.
- Primeira Lei: Um robô não pode ferir um ser humano ou, por omissão, permitir que um ser humano sofra algum mal.
- Segunda Lei: Um robô deve obedecer às ordens que lhe sejam dadas por seres humanos, exceto nos casos em que tais ordens contrariem a Primeira Lei.
- Terceira Lei: Um robô deve proteger sua própria existência desde que tal proteção não entre em conflito com a Primeira e Segunda Lei.

A construção de robôs começou a tomar força no início do século XX com a necessidade de aumentar a produtividade e melhorar a qualidade dos produtos. Em 1961, George Devol e Joe Engleberger desenvolvem o Unimates, que foi o primeiro robô industrial (JEFFERIS, 2006), e em 1966, o SRI começou a desenvolver Shakey (Figura 2.4), o primeiro robô móvel capaz de raciocinar sobre suas ações através de um planejador automático (STRIPS) (RUSSELL e NORVIG, 1995).

Na década de 80, ocorreu um grande progresso na área de arquitetura robótica móvel. Em 1986, Brooks apresenta a primeira arquitetura reativa, a arquitetura Subsumption, afirmando que “o mundo era seu melhor modelo” (BROOKS, 1986). E, em 1987, Arkin, apresenta a primeira arquitetura híbrida, a AuRA, afirmando que para um robô poder se locomover no ambiente dinâmico em que vivemos, seria necessário acoplar módulos reativos e deliberativos (ARKIN, 1998).

Figura 2.4 – Shakey, o primeiro robô móvel (WIKIMEDIA COMMONS, 2007).

Em 1997, a NASA envia o Mars Pathfinder com seu rover Sojourner para Marte (MISHKIN, 2003), reacendendo o imaginário das pessoas em relação à utilização de robôs móveis.

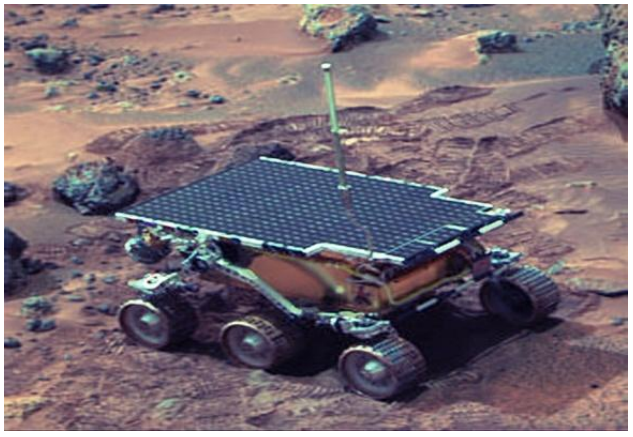


Figura 2.5 – O rover Sojourner da NASA em Marte (WIKIMEDIA COMMONS, 2005).

2.3 Plataformas Robóticas Móveis

Um robô móvel é um dispositivo autônomo capaz de se movimentar e interagir em um ambiente definido. Para esta finalidade, considera-se que um robô móvel é uma plataforma dotada de uma coleção de sensores, um sistema computacional embarcado e um

conjunto de atuadores. Sendo possível subdividir um robô móvel em duas partes: a Plataforma Robótica Móvel (parte mecânica) e o Controle Robótico (parte computacional).

A Plataforma Robótica Móvel é a estrutura física do robô, sendo composta por sensores, atuadores, componentes eletrônicos necessários e suas peças estruturais. Para construí-la é preciso considerar todos seus elementos, verificando seu correto posicionamento e funcionamento. Diversos trabalhos são apresentados na literatura apresentando a construção de uma Plataforma Robótica Móveis, dentre os quais podemos citar alguns trabalhos desenvolvidos no Laboratório de Automação Integrada e Robótica da Faculdade de Engenharia Mecânica da Unicamp (MELO, 2007; LIMA, 2003).

Atualmente existem inúmeras possibilidades de plataformas robóticas comerciais, com diferentes tamanhos, funcionalidades e preços. A Figura 2.6 apresenta alguns desses dispositivos, e a Tabela 2.1 apresenta um quadro comparativo das principais características desses dispositivos (processamento, sensores e atuadores).

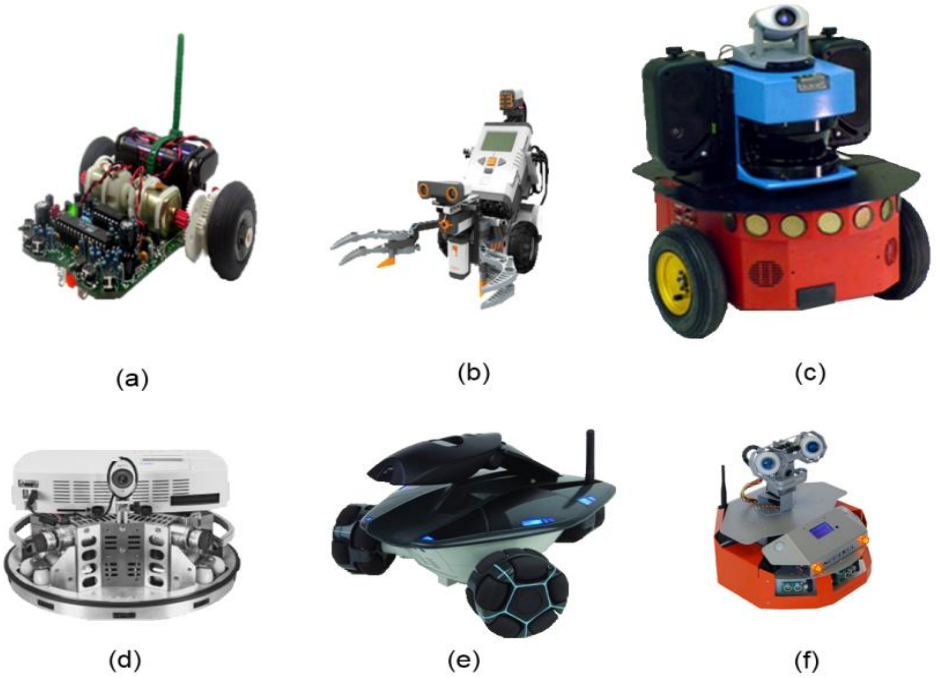


Figura 2.6 – Exemplos de Plataformas móveis comerciais. (a) ASURO (PICASA, 2008); (b) LEGO Mindstorm NXT (DEVICEGURU, 2008); (c) Pioneer 3-DX (ITMB, 2008); (d) Robotino (FESTO, 2008); (e) Rovio (GIZMO WATCH, 2008); (f) Sputnik (DRROBOT, 2008).

Tabela 2.1 – Principais Características de Plataformas móveis comerciais.

Plataforma	Processador	Sensores	Atuadores
ASURO	ATMEGA8L	Fototransistores para seguir trajetórias (2); Sensores de contato (6) Encoders de posição (2)	Motores DC (2)
Lego MindStorms NXT	ARM7TDMI ATmega48	Diversos (Ultrasom, Giroscópio, sensores de Toque e outros)	Servo-motor
Pioneer 3-DX	Renesas SH2-7144 (32 bits)	Sonares ultrasônicos (8) Laser Câmera Omnidirecional Compassos Gripper Encoders de posição (2) Wireless (opcional)	Motores DC (2)
Robotino	PC104	Sonares infravermelhos (8); Encoders de posição (3) Sensores de Contato Câmera Web USB Wireless	Motores DC (3)
Rovio	Marvell PXA270M	Encoders de posição (4) Câmera Microfone Wireless	Motores DC (4)
Sputnik	DSP	Encoders de posição (2) Câmeras (2) Sensores Ultrasônicos (3) Sensores Infravermelhos (3) Sensores de Contato Display LCD Wireless	Motores DC (2)

A seguir detalharemos informações sobre características de utilização desses dispositivos robóticos móveis apresentados na Figura 2.6:

(a) ASURO

O robô móvel ASURO foi desenvolvido com a finalidade de educação tecnológica pelo Instituto de Robótica e Mecatrônica do Centro Aeroespacial alemão. Trata-se de um robô simples, de baixo custo e flexível, com estrutura física aberta, utilizando como linguagem de programação C.

(b) LEGO MindStorms NXT

LEGO Mindstorms NXT foi lançado comercialmente em 2006, sendo uma linha de produtos da LEGO resultado de uma parceria entre o Media Lab do Massachusetts Institute of Technology (MIT) e o LEGO Group para a educação tecnológica. O conceito desse dispositivo é a arquitetura aberta para utilização com possibilidade de construir diversas plataformas diferentes com os mesmos blocos. Ele é constituído por um bloco programável NXT, sensores, servo-motores, mesa rotativa, bateria recarregável, conversor de energia, software de programação NXT e almoxarifado de peças para a implementação de dispositivos mecânicos (blocos, vigas, eixos, rodas, engrenagens e polias).

(c) Pioneer 3-DX

O P3-DX, comercializado pela MobileRobots Inc, é um robô muito utilizado pesquisas acadêmicas, referenciado muitas vezes em teses e artigos. Dentre as principais ferramentas disponibilizadas pelo fabricante, podemos destacar as seguintes:

- Condução através de um teclado ou joystick;
- Planejamento de trajetórias;
- Visualização de mapas com as leituras dos sonares e/ou dos lasers;
- Sistema de Localização através da utilizaçao de sonar;
- Programação em linguagem C/C++.

(d) Robotino

Robotino é uma plataforma robótica fabricada pela FESTO™ para educação tecnológica, treinamento e pesquisa. É uma plataforma aberta para inserção de diferentes sensores e atuadores, baseado em um conjunto de direção omnidirecional (três motores DC), que permite ao sistema movimentar-se livremente, sendo controlado por um sistema de PC com padrão industrial, capaz de planejar rotas de maneira totalmente autônoma. Este dispositivo é totalmente aberto, podendo ser programado no ambiente RobotinoView, desenvolvido em Labview, como também nas linguagens de programação C, C + +, Java,. NET, Matlab, Labview e Simulink.

(e) Rovio

Rovio é uma plataforma robótica móvel desenvolvida pela empresa WowWee, com capacidade de movimento omnidirecional (3 motores DC) e com câmera WEB integrada com possibilidade de movimentação (1 motor DC). Uma das principais características desse dispositivo é a incorporação da tecnologia Northstar, sistema semelhante ao sistema de micro-GPS residencial, permitindo assim que este dispositivo navegue pelo ambiente com precisão.

(f) Sputnik

Sputnik é uma plataforma robótica fabricada pela Dr. Robot Inc. Possui um controle de alto nível mantido por um PC/servidor, que pode ser local ou remoto, comunicando-se através de uma rede wireless. As funcionalidades de baixo nível são gerenciadas através de um processador de sinal digital integrado *onboard*.

2.4 Sensores

O controle de um dispositivo robótico móvel requer uma análise do sistema mecânico, de modo a descobrir a forma como o robô percebe o ambiente e como poderá

interagir com o mesmo, para isso é preciso entender o funcionamento dos sensores e atuadores disponíveis e analisar o modelo cinemático do robô móvel desenvolvido.

Na análise dos sensores é importante saber que num projeto de dispositivos robóticos móveis são utilizados dois tipos de sensores: os proprioceptivos, utilizados para a leitura de valores internos do sistema, como velocidade do motor, temperatura ou tensão de alimentação; e os exteroceptivos, utilizados para adquirir informações do ambiente, como a distância de um objeto, intensidade de luz ou indutância de um objeto.

Dentre os diversos tipos de sensores comerciais, cada um para um uso específico, entre eles, podemos destacar os seguintes:

- **Sensores de contato**

São geralmente constituídos de micro-chaves que retornam a condição de aberta ou fechada. Estes sensores são disponibilizados nos pára-choques dos robôs móveis, e assim, o robô obtém a informação de contato com algum objeto. Porém, abordagens mais sofisticadas utilizam sensores analógicos de forma a obter informações da força de contato exercida (MELO, 2007).

Normalmente, os sensores de contato são os últimos recursos utilizados para se evitar colisões. Em alguns sistemas, as informações provenientes dos sensores de contato são tratadas pelo controle do robô, entretanto na maioria dos robôs disponíveis no mercado, estas informações estão vinculadas diretamente na camada de nível mais baixo do controle (MELO, 2007).

- **Odometria óptica**

Os encoders óticos tornaram-se os dispositivos de uso mais geral para a leitura de posição e velocidade angular. Em robótica móvel são utilizados para controlar a posição e a velocidade das rodas ou outro atuador motorizado (SIEGWART e NOURBAKHSH, 2004).

Um encoder ótico (Figura 2.7) é composto de um dispositivo mecânico rotativo, uma fonte de luz e um receptor. O dispositivo mecânico possui diversas marcações, que podem ser perfurações ou marcas claras e escuras. Assim, como observado na Figura, 2.7 conforme a rotação do motor, o receptor recebe ou não a luz emitida, criando uma série de pulsos, e dessa forma, o robô conhece o movimento do motor.

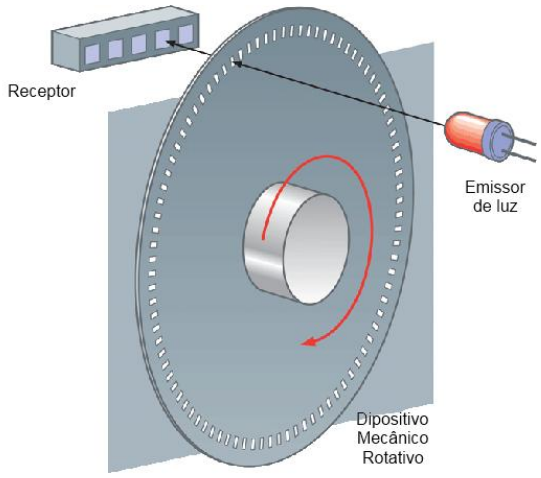


Figura 2.7 – Encoder Óptico (KALIPEDIA ROBOTICA, 2010).

- **Acelerômetros**

São dispositivos que procuram determinar a aceleração de um objeto considerando a 2ª Lei de Newton, devendo ser montados em dispositivos elásticos. Normalmente, cada acelerômetro mede apenas a aceleração em uma única direção, desta forma, para se adquirir a aceleração total de um objeto é necessário montar três acelerômetros, dispostos ortogonalmente um em relação ao outro, de forma a medir a aceleração nas direções *X*, *Y* e *Z*.

- **Ultrasom**

São utilizados para medir a distância de objetos através do eco criado pelo objeto, assim, são constituídos basicamente por um emissor sonoro e um receptor. O emissor envia um sinal sonoro que ecoa no objeto e retorna, sendo percebida pelo receptor, e assim, possibilitando o cálculo do tempo entre o envio do sinal e a recepção, ou o cálculo da

diferença de fase dos sinais. Conhecendo-se a velocidade do som (344m/s no ar), é possível calcular a distância do objeto através da equação 2.1, onde c é a velocidade do som e t é tempo de deslocamento encontrado (MELO, 2007).

$$d = \frac{c*t}{2} \tag{2.1}$$

Uma das dificuldades na sua utilização deve-se ao fato que o alcance deste método ser muito curto, limitando sua utilização. Além disso, existem algumas matérias que absorvem a energia sonora, diminuindo bruscamente a amplitude do eco, impossibilitando sua detecção.

- **Laser**

Os sensores laser operam de forma similar os sensores de ultrasom, a partir da emissão de um sinal de luz, e determinação da distância de um objeto a partir da medição do tempo de reflexão desta luz pelo objeto.

Os sensores laser detectam distâncias maiores que os sensores de ultrasom por causa da menor dispersão do sinal, tornando este sistema mais eficaz, contudo, devido à velocidade da luz, é necessário um processamento muito mais rápido, o que torna este sistema, também, mais caro (SIEGWART e NOURBAKHSH, 2004).

- **GPS**

O sistema GPS consiste na obtenção de informações de diversos satélites distribuídos ao redor da Terra que emitem dados de sua localização e tempo atual e um ou mais receptores no robô móvel. Desta forma, a partir, destas informações, o robô encontra sua localização através da técnica de triangulação da posição dos satélites.

Com a utilização de satélites, que possuem distâncias diferentes, o tempo entre ao envio dos dados e a recepção pelo robô pode variar razoavelmente, assim, faz-se necessário a utilização de 4 satélites de maneira a encontrar as 4 variáveis: *X*, *Y*, *Z* e o tempo atual. Uma restrição na utilização do GPS é a impossibilidade de sua utilização na maioria dos ambientes internos, pois o robô não é capaz de receber os sinais dos satélites.

2.5 Cinemática de Robôs Móveis

O estudo da cinemática de um robô tem como objetivo descobrir suas características e restrições mecânicas, de maneira a encontrar as possíveis velocidade e posições no espaço que um robô pode atingir. E, dessa forma, conhecendo o modelo cinemático do robô, planejar as ações que o robô deve realizar para alcançar seu objetivo. Em robótica móvel, é necessário entender o comportamento mecânico do robô tanto para o projeto de robôs móveis quanto para o desenvolvimento de um software de controle para um robô móvel (SIEGWART e NOURBAKHSH, 2004).

Um conceito importante na cinemática de um robô é a sua holonomicidade, a qual indica a existência ou não de restrições nos seus movimentos do robô, dividindo os robôs em duas categorias, os robôs holonômicos que podem se mover em qualquer direção a partir de qualquer posição; e os robôs não-holonômicos que possuem restrição em pelo menos uma direção, que podem ser provocadas pela conservação do momento angular, pela restrição de não-deslize de uma roda, pelo fato de o sistema não ter atuadores em todas as direções do espaço do problema, ou ainda por outros fatores (MURRAY, 1994). Contudo, é necessário salientar que apesar de seus movimentos serem limitados, os robôs não-holonômicos podem atingir qualquer configuração no espaço.

No planejamento, além do modelo cinemático deve-se considerar também o espaço de trabalho do robô. O espaço de trabalho é o conjunto das possíveis posições que um robô móvel pode atingir em seu ambiente, ou seja, os locais acessíveis ao robô (SIEGWART e NOURBAKHSH, 2004).

O procedimento para obter um modelo para todas as possibilidades de movimentação do robô móvel começa com a compreensão do método de locomoção utilizado pelo robô, onde cada robô pode possuir um método diferente em função de seus diferentes atuadores locomotores (normalmente pernas ou rodas). Cada atuador contribui individualmente para o movimento do robô, como também impõe restrições nos movimentos (e.g., movimentos laterais para robôs móveis com direção diferencial).

Neste trabalho os atuadores locomotores são as rodas do robô, portanto, para poder controlar o robô móvel, é necessário entender como cada roda contribui para a movimentação do robô e quais as restrições que a geometria das rodas implica na movimentação do robô.

2.5.1 Geometria da rodas

A roda tem sido incontestavelmente, o mecanismo de locomoção mais popular em robótica móvel e em veículos construído pelo homem. Podendo atingir altos ganhos de eficiência, e faz isso com uma implementação mecânica relativamente simples sistema, também, mais caro.

Existem quatro classes principais de roda: (a) roda padrão, (b) roda caster, (c) roda sueca, (d) roda esférica (Figura 2.8). Elas diferem muito em sua cinemática e, portanto, a escolha do tipo de roda tem um grande efeito sobre a cinemática global do robô móvel (SIEGWART e NOURBAKHSH, 2004).

A roda padrão (Figura 2.8a) e a roda caster (Figura 2.8b) têm um eixo principal de rotação, sendo assim, altamente direcional. Para mover em uma direção diferente, a roda deve ser primeiramente orientada ao longo de um eixo vertical. A principal diferença entre estas duas rodas é que a roda padrão pode realizar este movimento de orientação, sem efeitos colaterais, como o centro de rotação atravessa a área de contacto com o solo, enquanto a roda caster gira em torno de um eixo de deslocamento, transmitindo uma força à base do robô durante sua orientação (SIEGWART e NOURBAKHSH, 2004).

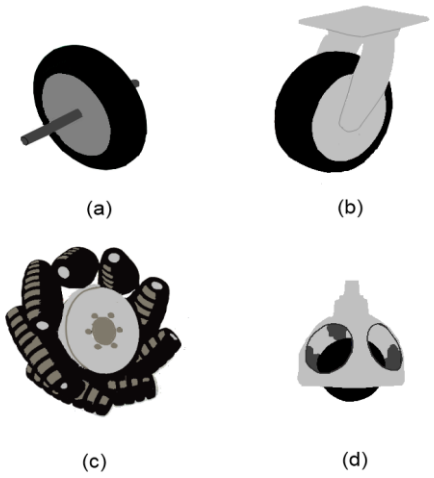


Figura 2.8 – Classes de roda. (a) roda padrão; (b) roda caster; (c) roda sueca; (d) roda esférica.

A roda sueca (Figura 2.8c) e a roda esférica (Figura 2.8d) são as rodas que menos afetam os projetos. A roda sueca funciona como a roda padrão, mas apresenta baixa resistência em outra direção, podendo ser perpendicular à direção convencional (roda sueca de 90º) ou em um ângulo intermediário (roda sueca de 45º). São utilizados roletes passivos em torno da circunferência da roda e o eixo principal da roda serve como a única alimentada ativamente comum.

A roda esférica é verdadeiramente uma roda omnidirecional, muitas vezes concebida de modo que possa ser alimentada ativamente para girar ao longo de qualquer direção. O mecanismo utilizado no desenvolvimento deste projeto é similar ao mouse de um computador (SIEGWART e NOURBAKHSH, 2004).

A escolha do tipo de rodas de um robô móvel está fortemente ligada à sua geometria, que possui um grande número de variações possíveis, como mostra os exemplos de aplicações mais utilizados apresentados na Figura 2.9.

Apesar das variações, existem importantes tendências e agrupamentos que podem auxiliar a escolha das rodas e sua geometria através da compreensão das vantagens e

desvantagens de cada configuração. Três características fundamentais de um robô para esta compreensão são: estabilidade, manobrabilidade e controlabilidade.

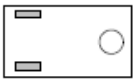
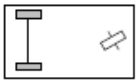
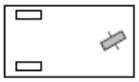
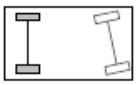
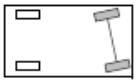
Geometria das rodas	Exemplos
	bicicletas, motos
	Cye
	Pygmalion, Alice
	Mini caminhões Piaggio
	Neptune, Hero-I
	Tribolo
	Denning MRV-2
	Carros com tração traseira
	Carros com tração dianteira

Figura 2.9- Possíveis disposições das rodas.

- **Estabilidade**

A estabilidade estática requer um mínimo de três rodas, e a condição que o centro de gravidade deve ser contido dentro do triângulo formado pelos pontos de contato do solo das rodas. É possível alcançar a estabilidade estática com apenas duas rodas se o centro de massa estiver abaixo do eixo da roda, contudo, em circunstâncias normais tal solução requer diâmetros de roda muito grandes (SIEGWART e NOURBAKHSH, 2004).

A estabilidade pode ser melhorada pela adição de mais rodas, embora uma vez que o número de pontos de contato superior a três, a natureza da geometria hiperestática exigirá a utilização em terrenos irregulares de um sistema de suspensão flexível. (SIEGWART e NOURBAKHSH, 2004)

- **Manobrabilidade**

A manobrabilidade de um robô é a sua capacidade de se movimentar em qualquer direção dentro de um ambiente, ou seja, é a facilidade que o robô possui para se movimentar.

Alguns robôs são omnidirecional, o que significa que eles podem se movimentar em qualquer direção, independentemente da orientação do robô, ou seja, estes robôs são designados como robôs holonômicos. Este nível de manobrabilidade exige rodas que possam se mover em mais do que apenas uma direção, de forma independente e sem restrições.

O grau de manobrabilidade de um robô inclui os graus de liberdade que o robô utiliza diretamente através das velocidades das rodas, e os indiretamente utilizados para alterar a configuração de direção e movimento.

- **Controlabilidade**

Existe geralmente uma correlação inversa entre controlabilidade e manobrabilidade. Assim, os projetos de robôs omnidirecionais exigem um alto processamento para converter as velocidades de rotação e translação desejadas para comandos individuais das rodas. Além disso, os robôs omnidirecionais freqüentemente têm maiores graus de liberdade nas rodas, causando um acúmulo de derrapagem, o que tende a reduzir a precisão no cálculo de posição e aumentar a complexidade do projeto.

Podemos afirmar que não existe uma configuração geométrica ideal para as rodas, que simultaneamente maximize a estabilidade, a manobrabilidade e a controlabilidade. Cada configuração impõe restrições diferentes, e a tarefa do projetista é escolher a configuração mais adequada possível para alcançar os objetivos.

2.5.2 Postura do robô

Na análise cinemática o robô é considerado como um corpo rígido em rodas, operando no plano horizontal. (SIEGWART e NOURBAKHSH, 2004). Assim, para especificar a posição do robô (Figura 2.10), é preciso estabelecer a relação entre o frame de referência local do robô e o frame de referência global.

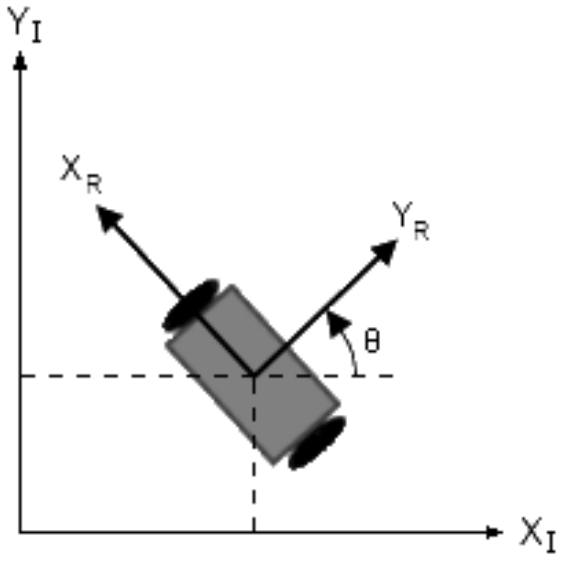


Figura 2.10 – Postura de um robô com direção diferencial.

A equação de postura do robô (equação 2.2) é determinada por três grandezas, duas para o seu posicionamento num plano (x,y) e outra para a orientação do robô em relação ao frame global. O índice I indica que esta postura esta referenciada ao frame global.

$$\xi_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{2.2}$$

2.5.3 Cinemática Direta

O objetivo da cinemática direta é obter a posição e velocidade do robô em função da velocidade de seus atuadores, neste caso, as suas rodas. Para calcular o deslocamento do robô em relação ao frame global é utilizada a equação 2.3.

$$\dot{\xi}_I = R^{-1}(\theta) * \dot{\xi}_R \tag{2.3}$$

Onde $R^{-1}(\theta)$ é a matriz rotação (equação 2.4) considerando a posição inicial θ e ξR é o deslocamento do robô em relação ao frame local (equação 2.5).

$$R(\theta) = \begin{bmatrix} cos\ (\theta) & sen\ (\theta) & 0 \\ -sen\ (\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

$$\dot{\xi}_R = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{2.5}$$

2.6 Descrição da Plataforma Robótica utilizada

Neste trabalho utilizaremos o dispositivo robótico móvel ASURO (Another Small and Unique Robot from Oberpfaffenhofen) desenvolvido pelo Instituto de Robótica e Mecatrônica do Centro Aeroespacial Alemão (DLR), disponível no Laboratório de Automação Integrada e Robótica da Faculdade de Engenharia Mecânica da UNICAMP.

O ASURO foi desenvolvido de forma a ser um robô simples e flexível, com estrutura física aberta, baseado em direção diferencial com um terceiro ponto de apoio para fornecer equilíbrio estático, podendo ser programado em qualquer linguagem, com suporte pra Windows e Linux. Ele é vendido comercialmente sob a forma de um kit, que possui componentes eletrônicos disponíveis comercialmente.

Ele é composto de um processador ATMEGA8, dois fototransistores seguidores de linha, seis sensores de contato, dois motores DC com encoders e sensores infravermelhos para comunicação. A Figura 2.11 apresenta o robô ASURO e seus principais componentes.

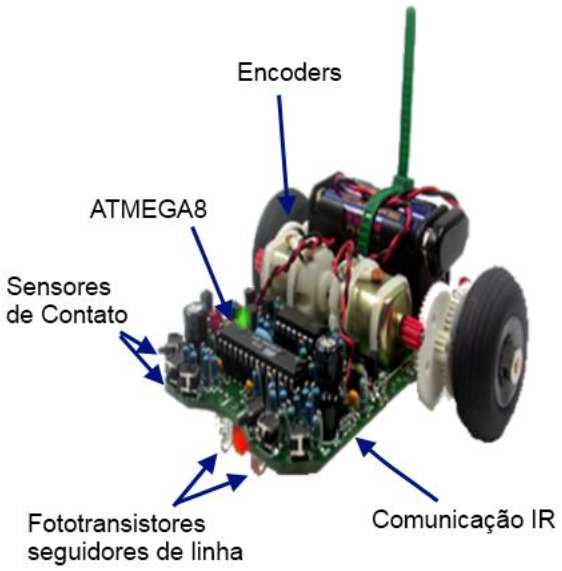


Figura 2.11 – Principais componentes do robô ASURO.

Na execução deste trabalho foi necessário observar algumas as restrições do ASURO, como a baixa precisão dos encoders e o não funcionamento dos motores para baixas velocidades, onde os motores não giravam quando solicitada uma velocidade entre 0 a 5cm/s.

Toda a implementação computacional desenvolvida nesse trabalho de pesquisa foi realizada utilizando o programa WinAVRC no ambiente Windows, este programa é fornecido pelo fabricante no CD de instalação e pelo site do distribuidor. O programa é transmitido para o processador do ASURO através de interface serial RS231 ou USB (FLASH). Ambos os programas estão disponibilizados para download no site WEB do fabricante.

Atualmente, existe uma grande comunidade de desenvolvimento de programas para o ASURO, disponibilizando aos usuários deste dispositivo robótico diversas bibliotecas de programas livres (ASUROWIKI, 2009).

2.7 Cinemática de um robô com direção diferencial

O dispositivo robótico móvel ASURO é um sistema de direção diferencial, conseqüentemente a análise cinemática abordada nesta seção será relacionada a robôs com direção diferencial, baseada no livro de Dudek e Jenkin (2000).

No caso da direção diferencial o movimento do robô é determinado apenas pelas velocidades das rodas, onde o robô circunda em torno de um CCI, ponto em que se localiza o eixo comum das duas rodas para uma curva desejada. Variando a velocidade relativa das duas rodas, o ponto CCI modifica sua posição, alterando a trajetória.

A Figura 2.12 mostra o funcionamento de uma direção diferencial, ilustrando os parâmetros utilizados, onde l é a distância entre as rodas, r é o raio de cada roda, V_D e V_E são respectivamente, as velocidades lineares das rodas direita e esquerda, R é o raio de curvatura da trajetória e ω é a velocidade angular do robô.

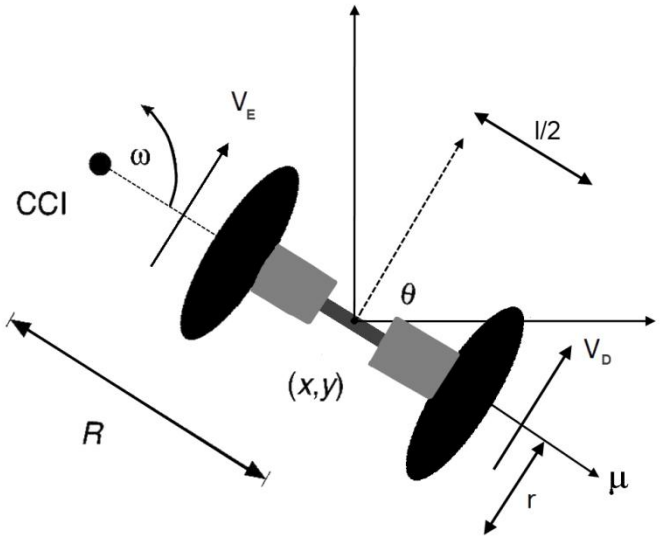


Figura 2.12 – Esquema de operação de uma direção diferencial.

Considerando a Figura 2.8, as velocidades lineares V_D e V_E são encontradas a partir das equações (2.6) e (2.7).

$$V_D = \omega * \left(R + \frac{l}{2} \right) \tag{2.6}$$

$$V_E = \omega * \left(R - \frac{l}{2} \right) \tag{2.7}$$

Há alguns casos particulares para o cálculo das velocidades, quando o robô precisar se locomover em linha reta, o raio R é infinito e ω é zero, para este fim, considera-se que $V_D = V_E$. Outro caso especial é comprovado pelas equações, quando o robô precisar girar em torno do seu eixo, o raio R=0, assim $V_D = -V_E$.

A partir das equações (2.6) e (2.7), a velocidade linear do robô V, a velocidade angular ω e o raio R, são obtidos através das equações (2.8), (2.9) e (2.10).

$$V = \left(\frac{V_E + V_D}{2} \right) \tag{2.8}$$

$$R = \frac{l}{2} * \left(\frac{V_E + V_D}{V_E - V_D} \right) \tag{2.9}$$

$$\omega = \frac{V_D - V_E}{l} \tag{2.10}$$

Considerando que as velocidades angulares de cada roda, ω_D e ω_E, podem ser obtidas através das equações (2.11) e (2.12), e, utilizando as equações (2.6) à (2.10), determina-se a equação da matriz do modelo cinemático (equação 2.13).

$$\omega_D = \frac{V_D}{r} \tag{2.11}$$

$$\omega_E = \frac{V_E}{r} \tag{2.12}$$

$$\begin{bmatrix} \dot{V}_X \\ \dot{V}_Y \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ \frac{-r}{l} & \frac{r}{l} \end{bmatrix} * \begin{bmatrix} \omega_E \\ \omega_D \end{bmatrix} \tag{2.13}$$

Portanto, com a matriz do modelo diferencial mostrada na equação (2.13) é possível encontrar o deslocamento do robô nos eixos *X* e *Y*. *E*, observa-se que a velocidade em *Y* é sempre zero, demonstrando a restrição holonômica da geometria das rodas em configuração diferencial. Esta restrição holonômica, demonstrada na equação (2.14), demonstra a impossibilidade do robô de transladar lateralmente, devendo efetuar diversos deslocamentos em *X* para poder alcançar uma posição lateral.

$$\mu = 0 \ \rightarrow \dot{x} * \text{sen}(\theta) - \dot{y} * \cos(\theta) = 0 \tag{2.14}$$

2.8 Conclusão

Neste capítulo foi realizado um trabalho de revisão bibliográfica aprofundado na área de Robótica Móvel, considerando a classificação de dispositivos robóticos móveis e a descrição das principais plataformas robóticas disponíveis no mercado e suas características em relação à modelagem, sensoriamento e principais utilizações.

Ainda neste capítulo foi enfatizada a descrição cinemática da plataforma robótica Móvel ASURO, disponível no Laboratório de Automação Integrada e Robótica, e utilizada como base de estudo experimental deste trabalho de pesquisa.

No próximo capítulo desta dissertação será desenvolvida uma proposta de sistema de navegação através do planejamento e execução de um caminho para atingir um objetivo específico, considerando-se o robô ASURO, e a arquitetura híbrida AuRA.

Para completar este trabalho de revisão bibliográfica recomenda-se a leitura dos anexos A, B, C e D dedicados a descrição detalhada de arquiteturas de controle, sistemas de localização, planejamento de movimento e navegação, que serão utilizados nos próximos capítulos deste trabalho.

CAPÍTULO 3

Proposta de Sistema de Navegação

A navegação robótica é o planejamento e execução de um caminho para atingir um objetivo específico, com possibilidade de uma adaptação da trajetória em função da necessidade (CROWLEY, 1984). O planejamento está associado à determinação das ações e movimentos do robô utilizando um modelo interno do mundo, e sua execução esta associada ao monitoramento das mudanças no ambiente e adaptação dos atuadores.

Para atingirmos os objetivos inicialmente delineados neste trabalho de pesquisa ter-se-á como base a plataforma robótica móvel ASURO, descrita anteriormente e a arquitetura de controle AuRA, assim, para detalhar cada módulo componente deste arquitetura, o presente capítulo está dividido em cinco etapas:

a) Percepção do ambiente,
b) Mapeamento e localização,
c) Planejamento do caminho,
d) Planejamento de trajetórias e
e) Execução de trajetória.

Como ressaltamos no capítulo de revisão bibliográfica, nos anexos A, B, C e D desta dissertação são apresentadas detalhadamente as abordagens utilizadas, explicando suas características e limitações.

3.1. Percepção do ambiente

A percepção do ambiente é obtida através da leitura e análise dos sinais dos sensores do dispositivo robótico móvel. Assim, para determinar a forma pela qual o robô irá perceber o ambiente, deve-se conhecer os sensores que o mesmo possui e determinar como estes sinais serão analisados.

O robô ASURO possui três tipos de sensores: sensores de contato, odometria e fototransistores, mostrados na Figura 3.1.

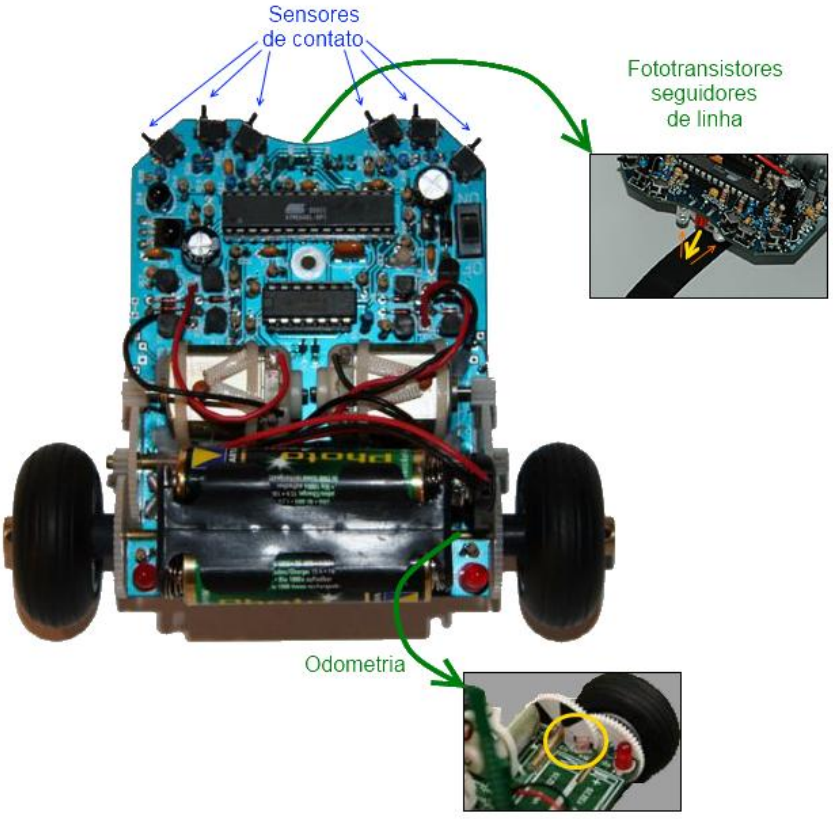


Figura 3.1 – Sensores do robô ASURO.

Os sensores de contato são utilizados na parte frontal do robô, detectando a existência de uma estrutura física na frente do robô, percebendo o ambiente e prevenindo possíveis danos ao dispositivo devido a obstáculos.

A odometria é realizada através de indicadores de rotação nas rodas (encoders óticos) utilizando dois fototransistores, um para cada roda, que funcionam de maneira a detectar a variação da cor branca para a cor preta no disco indicador disposto em cada roda. Assim, o número de mudanças de estado permite obter a rotação de cada roda, encontrando o deslocamento total do robô.

Os fototransistores conectados na parte inferior do dispositivo robótico móvel são utilizados para detectar a cor durante o deslocamento do robô num determinado caminho, permitindo seguir uma linha demarcada através da comparação entre as leituras dos dois fototransistores, designados de sensores seguidores de linha.

Assim, o robô ASURO navegará através do ambiente através dos sensores seguidores de linha e odometria, verificando sua permanência no caminho demarcado através dos fototransistores e calculando seu deslocamento utilizando a odometria. Os sensores de contato serão utilizados para detectar objetos durante o caminho, permitindo ao robô prevenir maiores danos.

3.2. Mapeamento e Localização

O mapeamento e a localização permitem orientar o robô em relação ao ambiente, situando-lhe e informando as posições do objetivo e dos objetos no ambiente. Assim, a forma que o ambiente é mapeado e o método utilizado para localizar-se alteram extremamente os comportamentos do robô, modificando o meio de percepção do mesmo e a abordagem de planejamento.

Neste trabalho, será utilizado mapeamento topológico devido sua correspondência com o estilo humano de localizar-se, e devido aos sensores existentes no robô ASURO a localização será efetuada através de um caminho demarcado na pista.

Os nós do mapa topológico serão atribuídos a locais onde haja interseção de dois caminhos e a lugares importantes e distintos. Por exemplo, numa planta fabril cada espaço de trabalho é representado por um nó, sendo o estoque um nó, a fundição outro nó, etc.

Para validação experimental desse trabalho utilizaremos o mapeamento de três ambientes de simulação distintos, apresentando seu mapa organizacional e topológico, e demonstrando com cada ambiente simulado foi mapeado. Os ambientes propostos foram os seguintes:

a) Fábril (Figura 3.2)
b) Escritório (Figura 3.4)
c) Sistema de Tubulação Industrial (Figura 3.5)

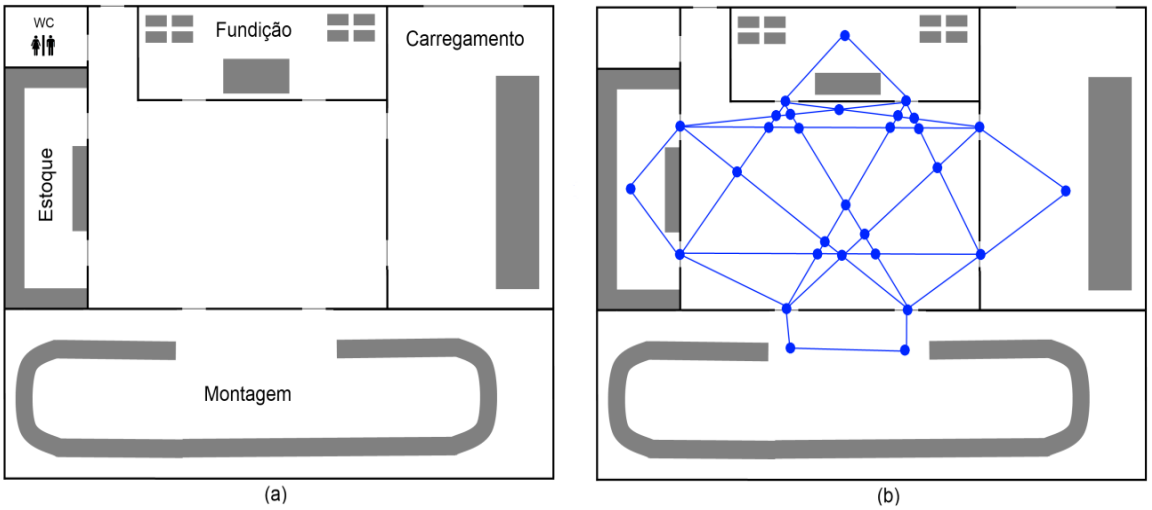


Figura 3.2 – Ambiente simulando uma fábrica. (a) Mapa organizacional. (b) Mapa Topológico

Considerando o grande número e dispersão dos nós, o mapa topológico de um ambiente fabril (Figura 3.2b) foi alterado para o mapa topológico apresentado na Figura 3.3, onde se utilizou uma trajetória diferente, conhecida como duplo infinito ou rosa de quatro pétalas, que possui um menor número de nós e uma melhor organização.

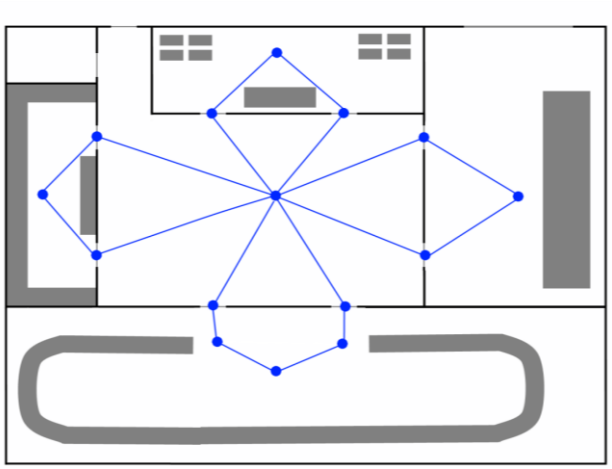


Figura 3.3 – Mapa Topológico da fábrica ajustado.

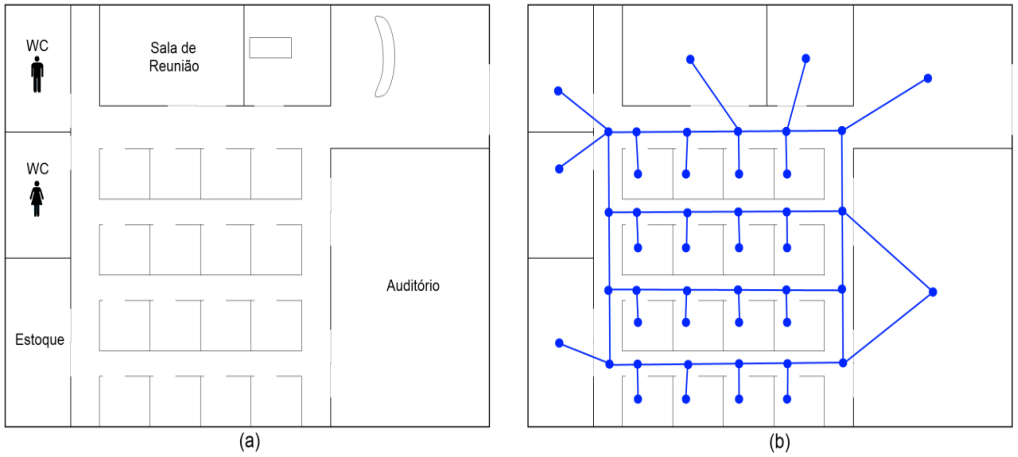


Figura 3.4 – Ambiente simulando um escritório. (a) Mapa organizacional. (b) Mapa Topológico

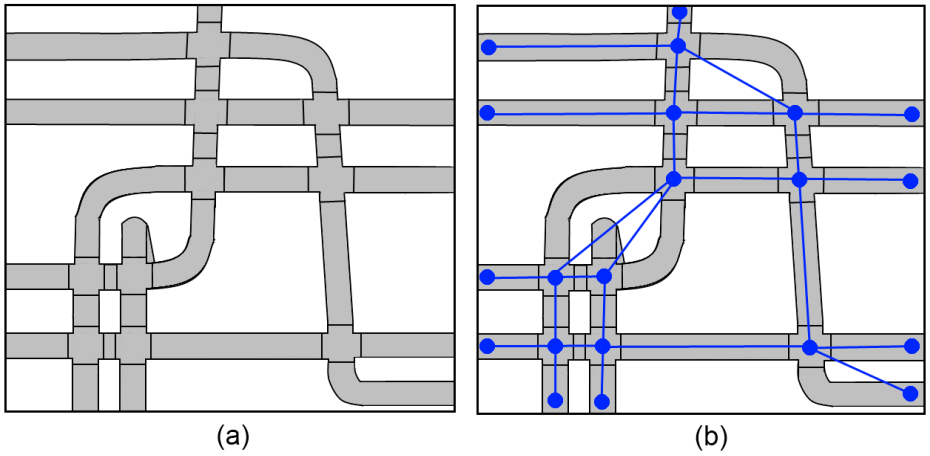


Figura 3.5 – Ambiente simulando um sistema de tubulação. (a) Mapa organizacional. (b) Mapa Topológico

Um grafo G = (N, E) representa o mapeamento topológico de um local, onde V são os nós do ambiente, e E são os trajetos que interligam estes nós.

Os nós V serão compostos pelos nós A, B, C, etc., e pelas coordenadas x e y de cada nó. Os trajetos E serão compostos pelo nó inicial e final, pelos pesos de ida e de volta, e pelos trajetos t1, t2, t3, etc.

Os pesos de ida e volta serão ajustados considerando a distância euclidiana entre os nós e o fator de segurança. O fator de segurança σ será atribuído conforme a facilidade de acesso da trajetória e a direção pretendida, pois mesmo os trajetos que são de duas vias, uma das direções será selecionada como preferencial, de forma a facilitar o fluxo caso haja mais robôs.

O fator de segurança σ é calculado pela equação (3.1), onde a facilidade de seguir a trajetória é representada por φ, que é determinado por um valor entre 0 e 0,5; e ν é o parâmetro de direção, onde ν recebe o valor 0 se for a direção organizacionalmente planejada ou recebe o valor 0,5 se for a direção oposta a planejada.

$$\sigma = \varphi + \nu \tag{3.1}$$

A distância euclidiana *d* é dada pela equação (3.2), onde o índice *i* indica o nó inicial e o *f* indica o nó final.

$$d = \sqrt{\left(x_i - x_f\right)^2 + \left(y_i - y_f\right)^2} \tag{3.2}$$

Calculados a distância euclidiana *d* e o fator de segurança σ, o peso ψ_{if} é determinado através da equação (3.3).

$$\psi_{if} = d * \sigma \tag{3.3}$$

Os trajetos t1, t2, t3, etc., deverão possuir pelo menos dois pontos (as coordenadas do nó inicial e final de cada trajeto), podendo ser adicionados mais pontos de controle, que poderão variar de acordo com a precisão estabelecida, porém, quanto mais pontos forem inseridos, maior será o tempo de processamento e de execução necessário.

A localização do robô será realizada através de rotas pré-determinadas e pela odometria. Cada ambiente terá uma pista demarcada, que passará pelos nós, onde serão colocadas marcas na pista utilizando uma cor diferente para informar ao robô a localização do nó. Entre um nó e outro, o robô irá estimar sua posição pela sua permanência ou não na pista demarcada e pela odometria, como apresentado na Figura 3.6, que demonstra a pista planejada para um ambiente fabril e ilustra a idéia das marcas dos nós.

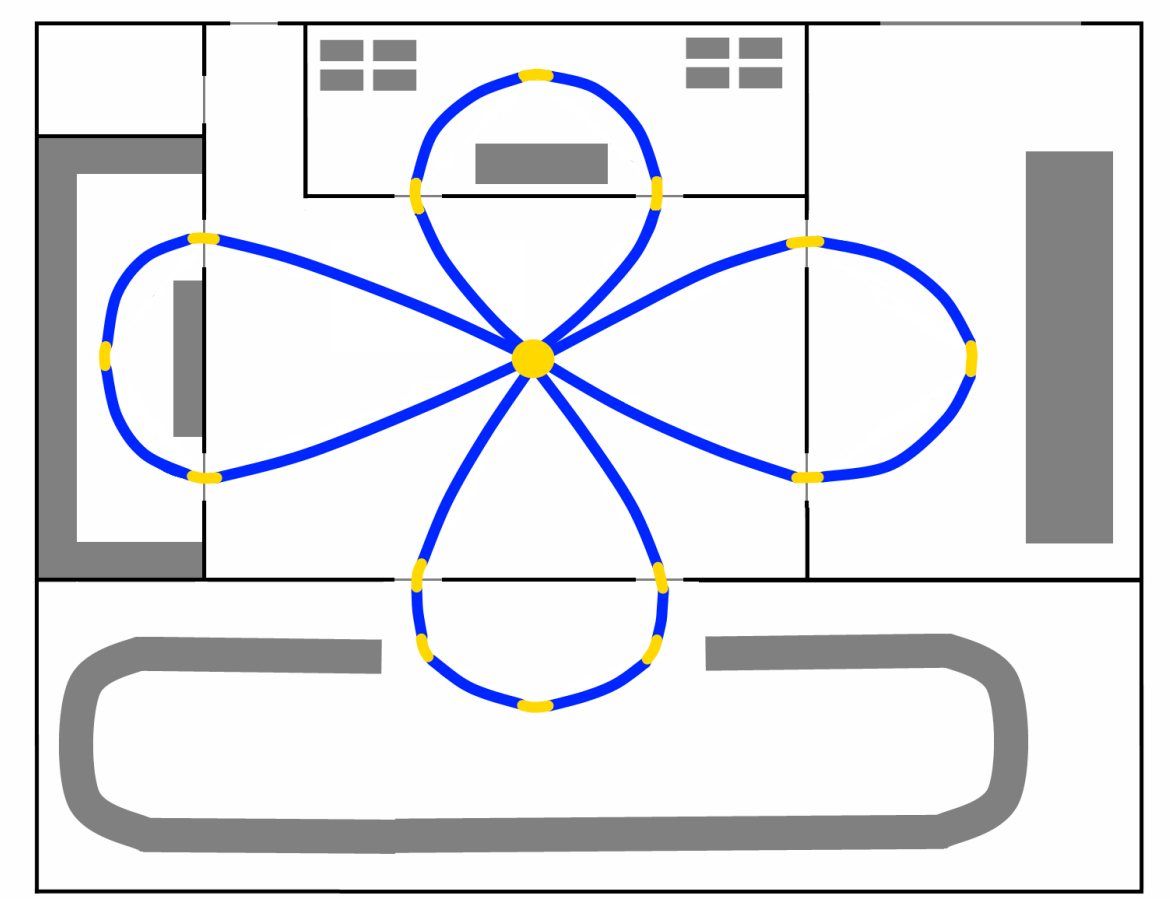


Figura 3.6 – Mapa da fábrica com a pista e os nós demarcados.

3.3. Planejamento do caminho

O planejamento de caminho é utilizado para fornecer os pontos pelos quais o robô deve passar. Para isso, o planejamento utiliza um algoritmo de busca para analisar o modelo interno do mundo e encontrar qual é o melhor caminho para seguir, resultando em uma seqüência de coordenadas a seguir sem colidir com os objetos conhecidos.

Com o propósito de determinar o melhor caminho a ser seguido, o planejamento do caminho utilizará uma tabela de reminiscência e o algoritmo de Dijkstra, apresentado no anexo C, de modo que a tabela de reminiscência é utilizada para arquivar e consultar os caminhos mais utilizados e o algoritmo de Dijkstra para determinar os nós pelo qual o robô deve passar na realização de um deslocamento desconhecido ou pouco utilizado através do cálculo de menor custo (algoritmo C.3). Portanto, inicialmente, no planejamento do caminho, o robô examina a tabela de reminiscência e verifica se o deslocamento desejado já foi calculado, em caso positivo, o robô verifica se o caminho está livre e utiliza-o, em caso negativo, caso haja uma obstrução no caminho arquivado ou caso o deslocamento desejado não esteja arquivado, o robô utiliza o algoritmo de Dijkstra para determinar o melhor caminho.

A tabela de reminiscência utiliza o conceito humano de selecionar um caminho para um deslocamento, pois os humanos geralmente utilizam o mesmo caminho para ir de um local a outro, apenas utilizando um mapa caso não conheça o local ou haja algum problema no caminho normalmente utilizado, como uma reforma na estrada. Portanto o robô armazenará 10 caminhos, com o ponto de partida, ponto de chegada, os nós utilizados e a freqüência semanal de uso, além disso, o robô terá uma lista com mais três caminhos freqüentemente utilizado na semana atual, de modo a atualizar a tabela de reminiscência.

No algoritmo de Dijkstra, considerando o nó em que o robô se encontra no momento do planejamento como nó inicial, primeiramente calcula-se o menor custo para chegar a cada nó, e depois de calculado todos os custos, o algoritmo determina o melhor

caminho, saindo do nó final até chegar o nó inicial através da observação do nó-pai de cada nó selecionado.

Para elucidar o procedimento do algoritmo de Dijkstra foi criado um exemplo ilustrado nas Figuras 3.7 - 3.13, onde a Figura 3.8 mostra o mapa topológico de um ambiente com os pesos de deslocamento e as demais Figuras mostram o procedimento de cálculo dos custos, considerando o nó *A* como o nó inicial, e a atribuição dos nós-pais de cada nó.

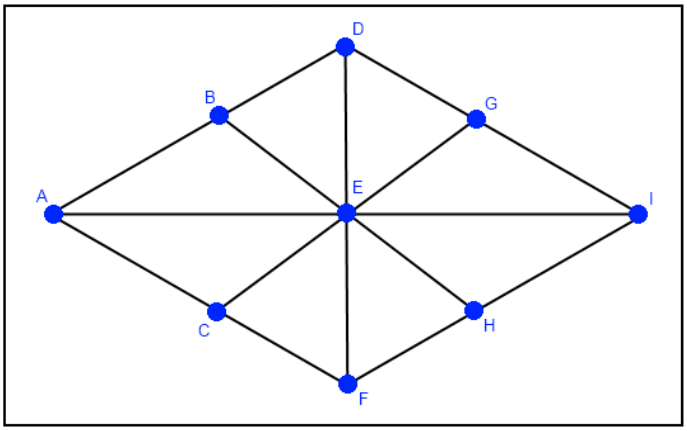


Figura 3.7 – Mapa topológico de um ambiente para a elucidação do algoritmo de Dijkstra.

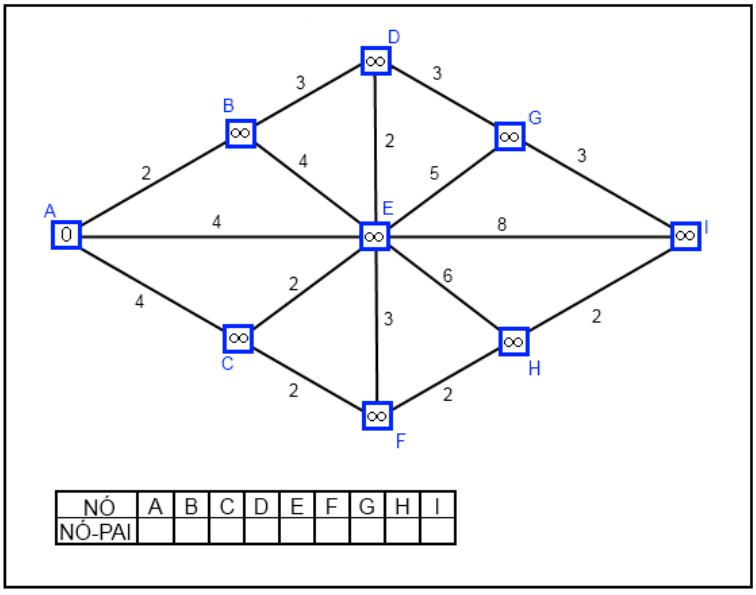


Figura 3.8 – Mapa topológico com os pesos de deslocamento.

Neste exemplo, os quadrados azuis representam os nós que não foram descobertos ainda, os losangos verdes representam os nós descobertos que permanecem em aberto, e os círculos cinzas representam os nós descobertos e fechados, ou seja, os nós que já foram analisados.

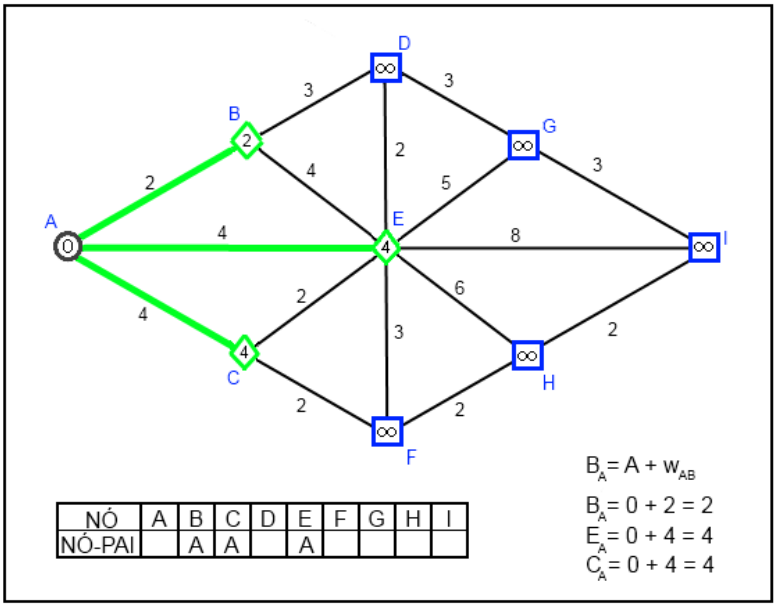


Figura 3.9 – Primeiro passo para p cálculo do melhor caminho.

No primeiro passo, ilustrado na Figura 3.9, o algoritmo analisa o nó inicial, descobre todos os nós que são conectados ao nó inicial, abrindo-os, calcula o custo de cada nó aberto e atribui o nó inicial como nó-pai de cada nó aberto, e por fim, fecha o nó inicial.

No segundo passo, ilustrado na Figura 3.10, o algoritmo escolhe o nó com o menor custo entre os nós abertos e analisa-o, descobrindo os nós fechados conectados a este nó, e calculando o novo custo para cada nó, se o custo diminuir, o custo do nó muda e o nó-pai muda também, mas se o custo aumentar ou permanecer o mesmo, nada se modifica, e por fim, o algoritmo fecha o nó analisado.

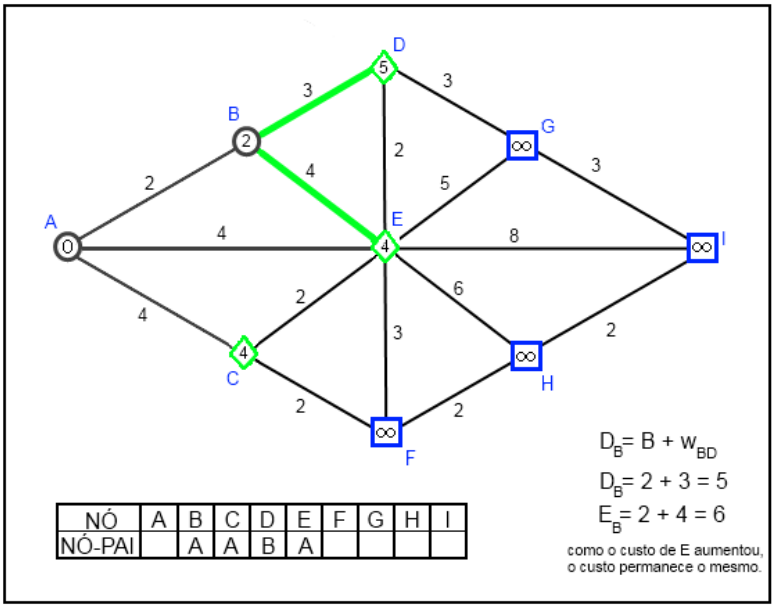


Figura 3.10 – Segundo passo para o cálculo do melhor caminho.

Este processo se repete até que todos os nós sejam fechados, de modo que o custo e o nó-pai de cada nó seja encontrado. Por fim, o algoritmo obtém uma configuração similar a encontrada na Figura 3.11.

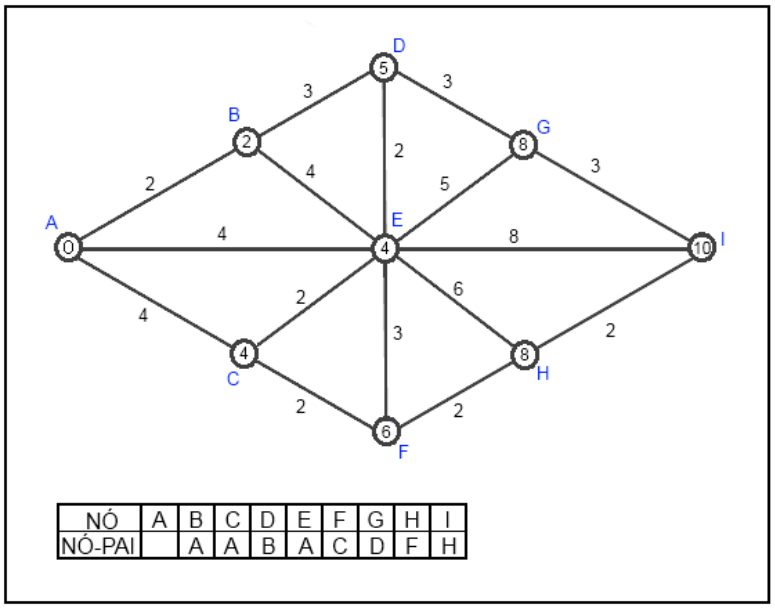


Figura 3.11 – Resultado do cálculo de melhor caminho

Após os resultados serem encontrados, como representado na Figura 3.11, o algoritmo procura pelo melhor caminho, e para isso é necessário verificar os nós-pais de cada nó desde o nó final até o nó inicial.

Considerando que o nó inicial é o nó A e o nó final é o nó I, as Figuras 3.12 e 3.13 ilustram esta parte do procedimento, onde na Figura 3.12, o algoritmo verifica qual é o nó-pai do nó final, neste caso é o nó H, e assim, o algoritmo marca o nó H.

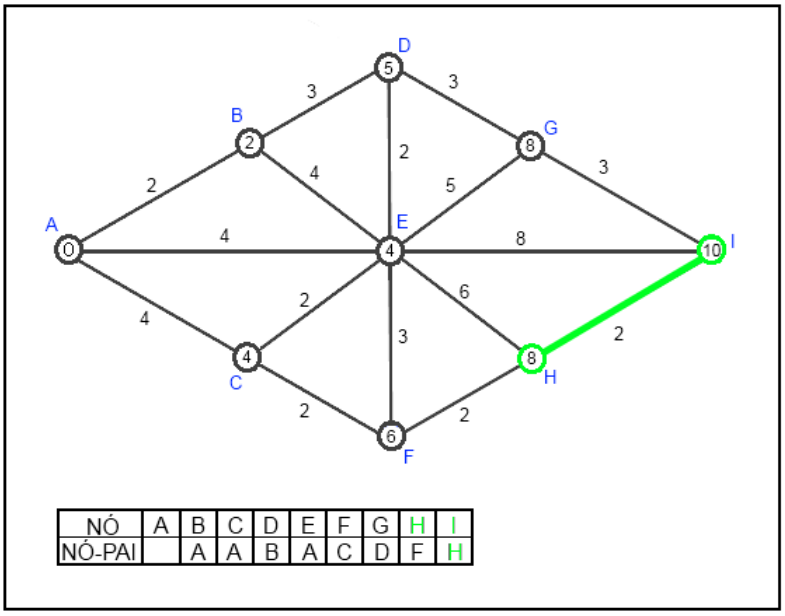


Figura 3.12 – Busca do trajeto de melhor caminho.

Depois, o algoritmo verifica qual é o nó-pai do nó H, e marca-o também. O algoritmo repete este processo até encontrar o nó inicial, o nó A. Após 5 passos, o algoritmo encontra o nó A, encontrando o melhor caminho entre o nó A e o nó I (A – C – F – H – I), mostrado na Figura 3.13.

Após a seleção do melhor caminho para o trajeto, pela tabela de reminiscência ou pelo algoritmo de Dijkstra, o robô determinará as coordenadas as serem atingidas através de um bando de dados, que dados pode ser calculado automaticamente através de algum algoritmo ou criado impositivamente pelo usuário. Se a precisão desejada for baixa, serão

utilizadas apenas as coordenadas do nós na ordem determinada, contudo, se a precisão necessária for maior, serão verificados os pontos de controle para cada trecho do caminho selecionado.

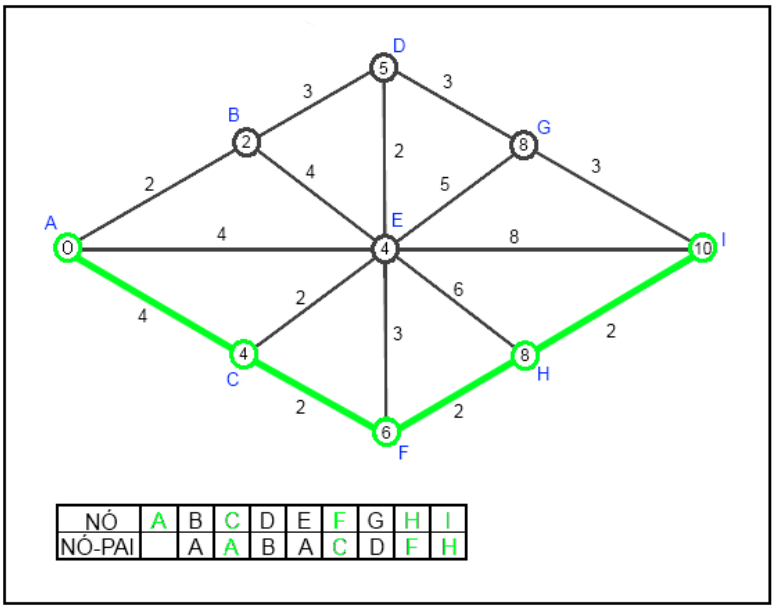


Figura 3.13 - Trajeto para o melhor caminho.

Portanto, resumindo, possuindo o mapa topológico do ambiente, e conhecendo sua posição atual e a do objetivo, o planejamento de caminho determina por quais nós o robô deve passar utilizando a tabela de reminiscência ou o algoritmo de Dijkstra, e então, transmite as coordenadas ao planejamento de trajetória utilizando um banco de dados.

3.4. Planejamento de trajetória

Nesta etapa, o robô utiliza as coordenadas selecionadas pelo planejamento de caminho para determinar os movimentos dos atuadores para que o robô atinja seu objetivo. Para esta tarefa existem duas abordagens: através de cálculos algébricos e geométricos ou por polinômios parametrizados.

Neste trabalho, foram utilizados os parâmetros do robô ASURO, e conseqüentemente os cálculos devem ser realizados para determinar às velocidades de ambas as rodas de um robô com direção diferencial. Além disso, devem ser consideradas as dimensões e as restrições não-holonômicas deste robô.

Para a execução do planejamento de trajetória pode-se destacar duas técnicas: caminho de Dubins (cálculos algébricos e geométricos) e β-splines (polinômios parametrizados), apresentadas detalhadamente no anexo C.

O caminho de Dubins tem como principal característica o cálculo do menor caminho entre as duas configurações, contudo, como o objetivo é seguir uma pista, o menor caminho pode se afastar razoavelmente da pista, como ilustrado na Figura 3.14, onde a linha em preto é a trajetória que deveria ser realizada, e a linha tracejada em azul é o caminho calculado pela técnica caminho de Dubins. Assim um meio de utilizar este método seria aumentar o número de pontos, entretanto, o tempo de processamento utilizando o caminho de Dubins é muito alto, pois são necessárias várias iterações matemáticas, e aumentar o número de pontos,para melhorar a precisão, aumentaria demasiadamente o tempo de processamento.

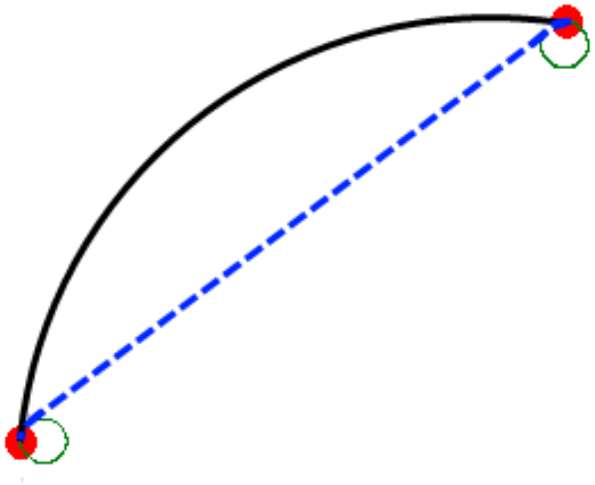


Figura 3.14 – Diferença entre a trajetória desejada e a calculada pelo caminho de Dubins.

O planejamento de trajetória através de β-splines pode alcançar uma grande precisão, pois tende a seguir precisamente a trajetória desejada, mas, para isso, é necessário um grande número de pontos, além disso, para executar a trajetória planejada através de β-splines é necessário possuir uma localização precisa do robô, algo difícil de obter com o robô ASURO, devido principalmente aos seus sensores. O grande número de pontos impõe uma mudança constante nas velocidades das rodas, o que torna a execução mais lenta e a hibridização da execução improvável, já que a verificação da localização e das velocidades ocorre constantemente, diminuindo o tempo entre uma percepção sensorial e outra.

Portanto, percebeu-se que ambas as técnicas possuem algumas características que as limitam, sendo consideradas insuficientes para suas utilizações neste trabalho, tornando necessário o estudo e implementação de outra técnica de planejamento.

Considerando a análise das técnicas e necessidades deste trabalho foi implementada uma técnica baseada no caminho de Dubins, que possibilita atingir o local desejado através de apenas um círculo, tornando-a mais rápida computacionalmente.

Esta técnica possui algumas restrições, dentre elas, a ausência de controle da orientação da configuração final e a possível abrangência, pois para alcançar uma trajetória admissível a posição final deve estar localizada dentro de uma faixa angular entre +45º e -45º ou entre +135º e -135º, visto que, numa faixa diferente, o robô alcançara o objetivo através de uma trajetória imprópria. A Figura 3.15 ilustra este conceito, onde o objetivo circular azul está localizado dentro do alcance imaginado, o mesmo não acontecendo com o objetivo losangular verde.

A maior limitação desta técnica é impossibilidade de controlar a orientação final, porém, aumentando o número de pontos obtém-se uma precisão satisfatória, e com baixo tempo de processamento. Ao mesmo tempo, como a trajetória é composta por curvas, e em cada uma delas, o robô utiliza uma velocidade constante em cada roda, esta técnica facilita a hibridização da execução.

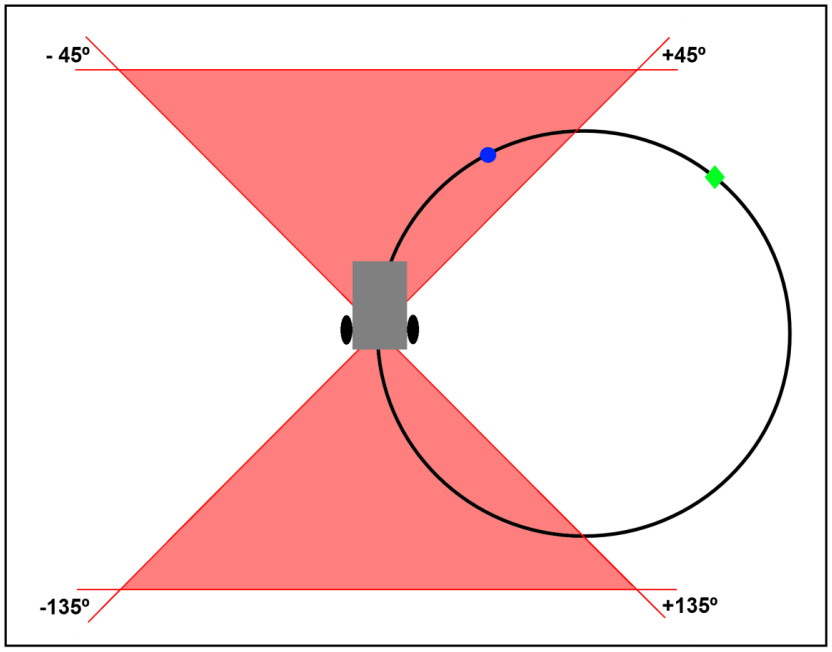


Figura 3.15 – Alcance da abordagem proposta.

Outro inconveniente é a localização de pontos fora da angulação desejada, como mostrado na Figura 3.15 (e.g. numa esquina), nestes casos, é possível rotacionar o robô ou utilizar pontos de auxílio para contornar este problema. Na primeira hipótese, o robô pararia no nó e rotacionaria até ficar alinhado com o próximo nó. No segundo caso, os pontos de auxílio seriam adicionados antes e depois do nó da aresta, de forma que o robô efetue uma curva neste logo, como ilustrado na Figura 3.16.

Portanto, apesar das limitações, utilizando um número razoável de pontos, esta técnica permite um planejamento rápido e uma execução precisa da curva, podendo seguir satisfatoriamente a pista demarcada.

Nesta técnica deve-se considerar a cinemática de robôs com movimentação diferencial, apresentada no capítulo 2. Assim, as velocidades lineares de cada roda serão obtidas através das equações (2.6) e (2.7). Portanto, para obtermos as velocidades lineares V_E e V_D é necessário conhecer o valor destas três variáveis: a velocidade angular do robô ω, o raio de curvatura R e a distância entre as rodas l.

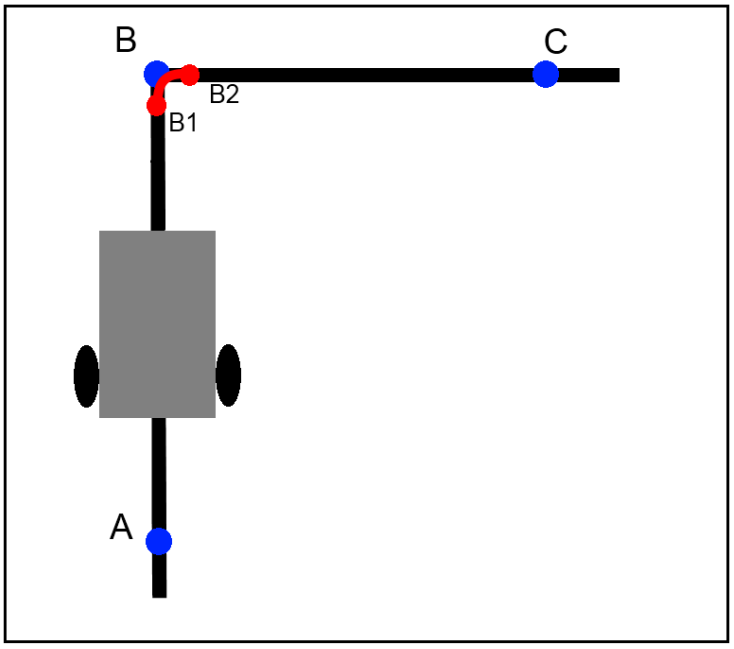


Figura 3.16 – Inserção dos pontos de auxílio numa curva aguda.

A distância entre as rodas *l* modifica-se a cada robô, porém é simples de se obter. A velocidade angular ω depende muito das restrições do robô e da resposta esperada. Neste trabalho, ω é diretamente proporcional ao raio de curvatura *R*, onde este é calculado através dos pontos inicial A e final B e da orientação inicial θ (Figura 3.17).

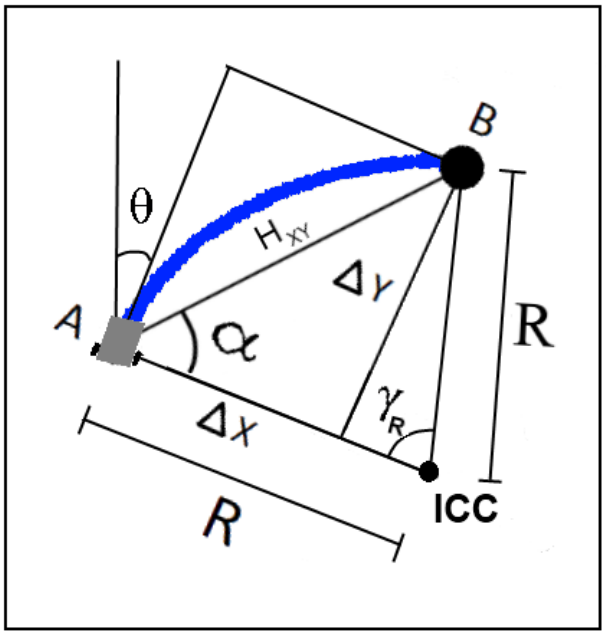


Figura 3.17 – Esquematização da abordagem de planejamento de trajetória.

Para determinar o valor do raio de curvatura *R*, primeiramente é necessário encontrar o ângulo α através da equação (3.4). Então, considerando que o triângulo formado pelos vértices A, B e ICC é isóscele, o ângulo de curvatura γ_R pode ser determinado através da equação (3.5).

$$\alpha = \tan\left(\frac{\Delta Y}{\Delta X}\right) \tag{3.4}$$

$$\gamma_R = 2 * \left(\frac{\pi}{2} - \alpha\right) \tag{3.5}$$

Com o valor de γ_R calculado, pode-se determinar o valor do raio *R* utilizando a equação (3.6).

$$R = \frac{\Delta Y}{sen(\gamma_R)} \tag{3.6}$$

Com a determinação do raio de curvatura *R* e a medida da distância entre as rodas *l*, só resta calcular a velocidade angular ω. Para este cálculo é necessário entender o conceito de velocidade angular, assim, sabe-se que velocidade angular é a velocidade em rad/s necessária para rotacionar um objeto em *G* graus num instante Δt, neste trabalho podemos, então, calcular *w* através da equação (3.7), onde Δt_d é tempo necessário para o robô se deslocar entre o ponto A e o ponto B.

$$\omega = \frac{\gamma_R}{\Delta t_d} \tag{3.7}$$

Deste modo, para determinar o valor de *w*, deve-se encontra o tempo de deslocamento Δt_d, que pode ser encontrado através da equação (3.8), onde Δd_d é a distância

a ser percorrida na trajetória entre A e B, e V_m é a velocidade linear média desejada do robô.

$$\Delta t_d = \frac{\Delta d_d}{V_m} \tag{3.8}$$

Para definir a distância Δd_d, deve-se considerar o deslocamento do centro de massa, portanto, Δd_d pode ser calculada através da equação (3.9). Porém, quando o raio de curvatura R for menor que metade da distância entre as rodas l, o deslocamento do centro de massa não vai corresponder com o deslocamento encontrado pela equação (3.9), pois neste caso as rodas efetuam um deslocamento maior que o deslocamento do centro de massa, portanto, nestes casos, o cálculo de Δd_d utilizará a equação (3.10).

$$\Delta d_d = \left|\gamma_R * (R)\right| \tag{3.9}$$

$$\Delta d_d = \left|\gamma_R * \left(R + \frac{l}{2}\right)\right| \tag{3.10}$$

A velocidade linear do robô V_m pode ser definida a partir das restrições do robô e dos parâmetros desejados na execução, e neste trabalho, deseja-se que a velocidade do robô varie entre 5 cm/s e 30 cm/s, sendo que a menor velocidade seja usada para curvas agudas, e a maior para curvas obtusas ou em linhas retas (R = ∞). Assim, considerando os requerimentos deste trabalho, a velocidade linear Vm será definida através da equação (3.11), sendo seu valor é dado em m/s.

$$V_m = 0{,}2 * \left|R\right| + 0{,}05 \tag{3.11}$$

Portanto, através da medida de l e das determinações dos valores de R e w utilizando as equações (3.4 - 3.11), as velocidades lineares das rodas podem ser calculadas através das equações (3.12) e (3.13).

$$V_D = \omega * \left(R + \frac{l}{2} \right) \tag{3.12}$$

$$V_E = \omega * \left(R - \frac{l}{2} \right) \tag{3.13}$$

Após o cálculo das velocidades V_D e V_E para cada trecho do caminho, os valores de V_D e V_E, juntamente com a distância de deslocamento necessária Δd_d em cada trecho, são transmitidas a parte da execução da trajetória.

Apesar desta abordagem possuir algumas limitações, a boa precisão e baixa necessidade de processamento tornam-na apropriada para este trabalho. Além disso, as limitações existentes foram superadas através de simples adaptações no planejamento.

3.5. Execução da trajetória

A etapa de execução da trajetória consiste no controle dos atuadores com base no planejamento da trajetória e na percepção do ambiente. Esta etapa pode ser realizada de forma puramente deliberativa, puramente reativa ou híbrida.

Neste trabalho, a execução da trajetória será realizada de forma híbrida, utilizando as velocidades calculadas no planejamento de trajetória como base e ajustando-as através da percepção do ambiente. Como este trabalho baseia-se na utilização da arquitetura AuRA, a etapa de execução de trajetória é realizada através de motor-schemas, detalhado no anexo A.

O método de motor-schemas, ilustrado na Figura A.7, é composto por três componentes distintos: esquemas perceptivos, esquemas motores (motor-schemas) e soma vetorial. Os esquemas perceptivos são um ou mais esquemas de processamento sensorial de um ou mais sensores. Neste trabalho, os esquemas perceptivos criados foram:

- **Percepção odométrica:** processamento da odometria relativa a cada roda, e estimativa de deslocamento total do robô;
- **Percepção de linha:** processamento da leitura dos fototransistores seguidores de linha;
- **Percepção de obstáculos:** processamento relativo aos seis sensores de contato na parte frontal do robô;
- **Percepção de estado:** analisa a resposta dos outros esquemas perceptivos a fim de verificar as condições do ambiente e do robô, de maneira a fornecer estas informações ao controlador de esquemas.

Os esquemas motores são os esquemas de processamento dos comportamentos, onde cada esquema (comportamento) gera um vetor resultante, baseando-se nas percepções do ambiente obtidas a partir dos esquemas perceptivos, que indica a direção que o robô deve seguir. Neste trabalho, ao invés de gerar um vetor, cada comportamento estabelece as velocidades para as rodas direita e esquerda. Os comportamentos implementados para a execução do robô neste trabalho são os seguintes:

- **Seguir o objetivo:** este comportamento procura manter a trajetória planejada de forma a alcançar o próximo objetivo, onde as respostas resultantes deste comportamento são as velocidades planejadas para cada trecho através do planejamento de trajetória. Assim, este comportamento é considerado puramente deliberativo;
- **Seguir uma linha:** este comportamento busca seguir a pista demarcada utilizando a percepção de linha, sendo puramente reativo;
- **Evitar obstáculos:** este comportamento utiliza a percepção de obstáculos para prevenir colisões entre o robô e o obstáculo;
- **Alerta:** este comportamento tem a função de detectar qualquer falha de funcionamento, como um obstáculo na pista ou o robô estar perdido. Nesta situação o robô pára os motores e sinaliza através dos leds, alertando ao usuário que o robô precisa de apoio pois encontra-se em dificuldade.

A soma vetorial soma as velocidades determinadas por cada comportamento, considerando o peso de cada comportamento, encontrando um velocidade final para cada roda. Os pesos dos comportamentos podem ser determinados pelo usuário ou encontrados automaticamente, sendo alterados pelo controlador de esquemas de acordo com os objetivos e devido à percepção do ambiente.

A Figura 3.18 ilustra a esquematização da parte da execução da trajetória, mostrando claramente a divisão desta etapa em quatro componentes: Esquemas Perceptivos, Esquemas Motores, Controlador de Esquemas e Soma Vetorial. Em seguida, cada componente constituinte da Execução da Trajetória será explicado, detalhando seus elementos principais.

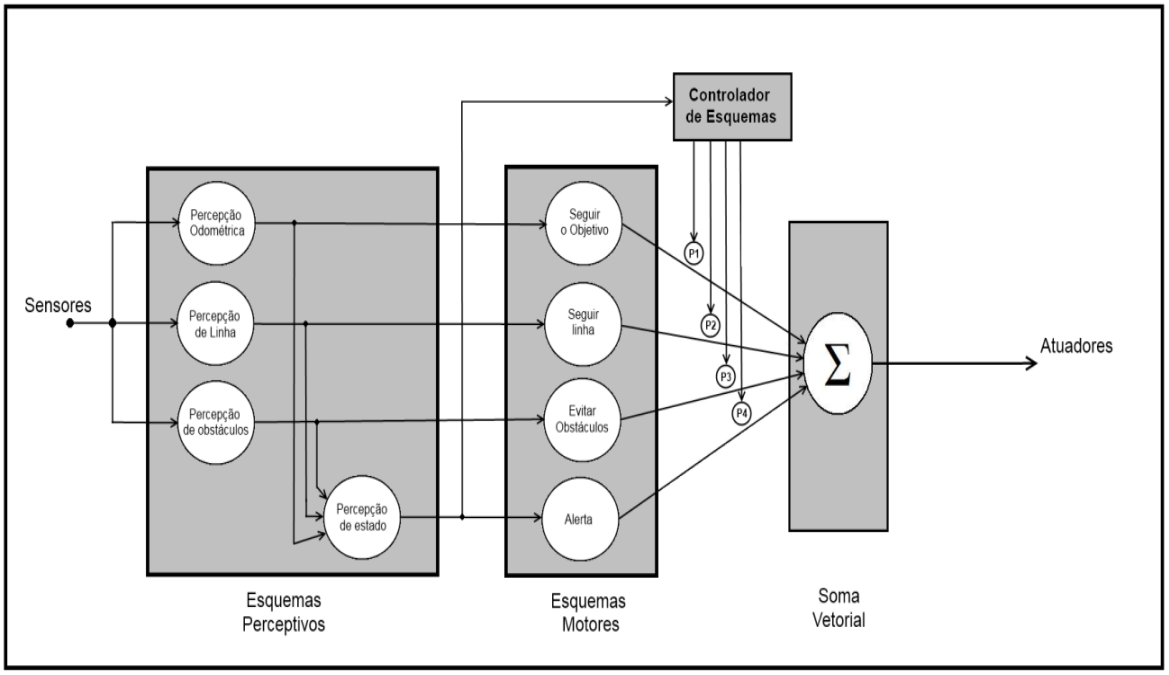


Figura 3.18 – Motor-schemas na execução da trajetória.

A percepção odométrica é responsável pelo cálculo do deslocamento do robô através da odometria, para isto, utilizar-se-á a equação (3.14), onde *DesR* é o quantidade de deslocamento do robô, $ODOM_D$ e $ODOM_E$ são as leituras obtidas pelos odômetros direito e esquerdo, respectivamente.

$$DesR = \frac{ODOM_D + ODOM_E}{2} \tag{3.14}$$

A percepção de linha determina a posição do robô em relação à pista utilizando as leituras dos fototransistores, de maneira a descobrir se ele está a à esquerda ou à direita do centro da pista. Para isso, é necessário obter a relação entre a diferença entre as leituras dos fototransistores e a posição do robô (distância ΔS entre a linha central da pista e a linha central do robô, ilustrada na Figura 3.19) através de experimentos.

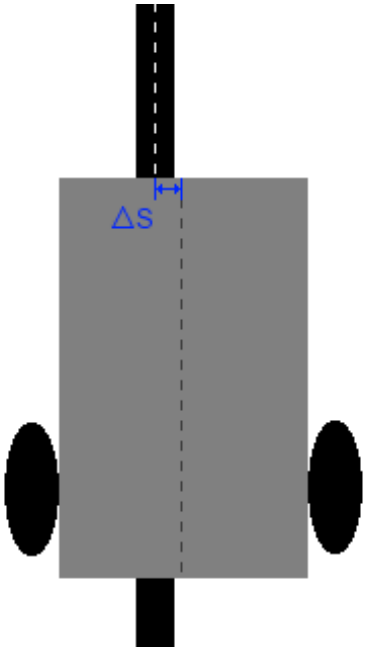


Figura 3.19 – Experimento para encontrar a variação a distância ΔS.

A percepção de obstáculo verificará a condição dos sensores de contato, e caso algum deles estiver sendo pressionado, este esquema sinaliza. A percepção de estado irá analisar a saída dos três esquemas perceptivos e classificar o estado como: caminho livre, caminho obstruído ou robô perdido.

O comportamento de seguir o objetivo utiliza a percepção odométrica para localizar-se no mapa, descobrindo em qual trecho está, e envia à soma vetorial as velocidades calculadas no planejamento de trajetória para trecho atual.

O comportamento de seguir linha receberá a informação da distância ΔS, determinado pela percepção de linha, e calculará as velocidades necessárias nas rodas para corrigir o desvio pelas equações (3.15) e (3.16), onde *largp* é a largura da linha, V_M é a velocidade máxima desejada no robô, K_R é o ganho reativo selecionado. Sabendo que a velocidade em ambas as rodas deve ser menor ou igual à V_M, após o cálculo, é necessário verificar e corrigir os valores encontrados.

$$V_D = V_M + K_R * \Delta S * V_M * \frac{2}{l \arg p} \tag{3.15}$$

$$V_E = V_M - K_R * \Delta S * V_M * \frac{2}{l \arg p} \tag{3.16}$$

Os esquemas de evitar obstáculos e alerta têm a responsabilidade de evitar maiores danos no robô caso este se perca ou o caminho estiver obstruído, e, em ambos os casos os dois esquemas serão acionados, onde o esquema de evitar obstáculo tem a responsabilidade de parar o robô, e o esquema alerta tem a responsabilidade de acionar os leds de forma a avisar ao usuário a situação do robô.

O controlador de esquemas permite obter a percepção de estado e modificar os pesos de cada comportamento de acordo com o estado atual. Estes pesos podem ser diferentes em função do robô e parâmetros do projeto, sendo obtidos através de análise e experimentos. Os pesos serão ajustados com base na Tabela 3.1, que será adequada conforme os resultados das simulações e experimentos.

Tabela 3.1 – Pesos dos comportamentos em cada estado.

	Caminho livre	Caminho obstruído	Robô perdido
P1	P_d	0	0
P2	P_r	0	0
P3	0	1	1
P4	0	1	1

Por fim, a soma vetorial, que controla os atuadores (rodas e leds), realiza a soma de todas as repostas dos comportamentos considerando o peso atribuído pelo controlador de esquemas, determinando as velocidades das rodas e o estado dos leds através das equações (3.17), (3.18) e (3.19), onde vd_i e ve_i são as velocidades determinadas em cada comportamento i, que varia de 1 a n; P_i é o peso do comportamento no estado atual, e led_i é o estado desejado para o led, sendo 1 para vermelho e 0 para verde.

$$V_D = \sum_{i=1}^{n} vd_i * P_i \tag{3.17}$$

$$V_E = \sum_{i=1}^{n} ve_i * P_i \tag{3.18}$$

$$LED = \sum_{i=1}^{n} led_i * P_i \tag{3.19}$$

3.6. Conclusão

Neste capítulo foi apresentado uma proposta de um sistema de navegação robótica através do planejamento e execução de um caminho para atingir um objetivo específico. Este sistema baseia-se na arquitetura AuRA, sendo dividido em cinco módulos: Percepção do ambiente; Mapeamento e localização; Planejamento do caminho; Planejamento de trajetórias; e Execução de trajetória.

A apresentação foi organizada de forma a detalhar cada uma dos cinco módulos separadamente. Em cada módulo foi utilizada uma abordagem, das quais pode-se destacar a utilização de mapeamento topológico, o algoritmo de Dijkstra e uma técnica baseada no caminho de Dubins.

No próximo capítulo desta dissertação serão apresentados os principais resultados experimentais obtidos em ambientes simulados e experimentos, mostrando a viabilidade da utilização do sistema de navegação proposto.

CAPÍTULO 4

Implementação Experimental e Resultados

Neste capítulo são apresentados os principais resultados obtidos em ambientes simulados e experimentais, permitindo demonstrar a viabilidade da utilização do sistema de navegação proposto. Todos os modelos implementados foram baseados na plataforma robótica móvel ASURO, disponível no Laboratório de Automação Integrada e Robótica da Faculdade de Engenharia Mecânica da UNICAMP.

Para a simulação computacional foi utilizado o aplicativo DD&GP - Differential Drive and Global Positioning, desenvolvido em ambiente MATLAB-Simulink® por Jahanzeb Rajput.

4.1. Simulador de Robôs Diferenciais

A biblioteca (blockset) MATLAB-Simulink® DD&GP é um ambiente de simulação para modelagem dinâmica e controle de sistemas robóticos móveis de direção diferencial, desenvolvido a partir da interface GUI (Graphical User Interface) do ambiente MATLAB®, que permite a construção e simulação de posicionamento desta classe de robôs móveis dentro de um ambiente de atuação.

Este blockset é constituído de sete blocos funcionais: PMDC Motor, Gearbox, Motor Driver, Mechanical Dimensions, Positioning, Dual PID, e o bloco Animation. A integração destes blocos funcionais permitirá a simulação do dispositivo robótico móvel diferencial em estudo a partir de suas especificações tecnológicas e funcionais. A Figura 4.1 mostra a montagem de uma simulação utilizando este aplicativo.

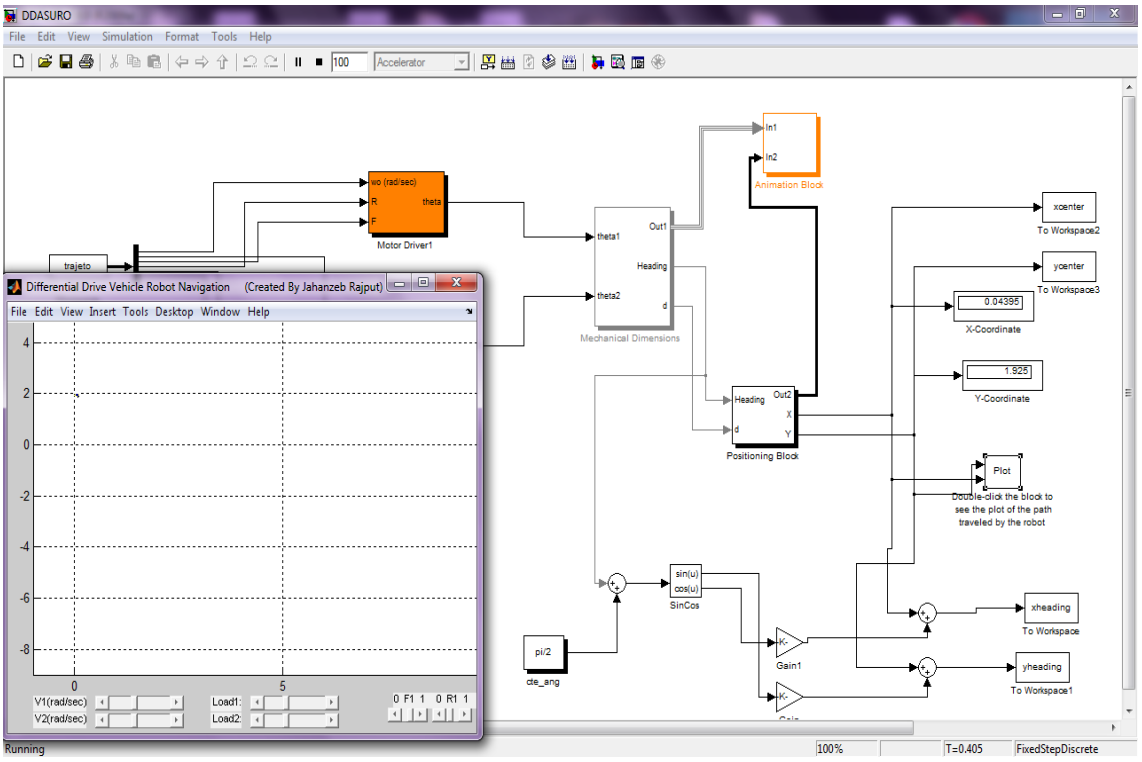


Figura 4.1 – Montagem da simulação utilizando o blockset DD&GP.

Como foi descrito anteriormente, o sistema de navegação proposto foi avaliado a partir da simulação em três ambientes diferentes: ambiente fabril, escritório administrativo e uma rede de tubulação industrial, ilustrados na Figura 4.2.

Para validação de resultados, a simulação computacional considera a realização de dois eventos distintos a serem realizados pelo dispositivo robótico móvel:

a) Realização de tarefa de inspeção num determinado trecho ou em todo ambiente, onde a trajetória será especificada ao robô sem um planejamento de caminho;

b) Deslocamento entre dois locais distintos do ambiente com a finalidade de transportar uma peça, objeto ou ferramenta. Neste caso, o robô realizará todo o planejamento, de forma a se deslocar num determinado local do ambiente, utilizando um mapa topológico e um banco de dados contendo as coordenadas dos locais importantes e dos pontos de controle.

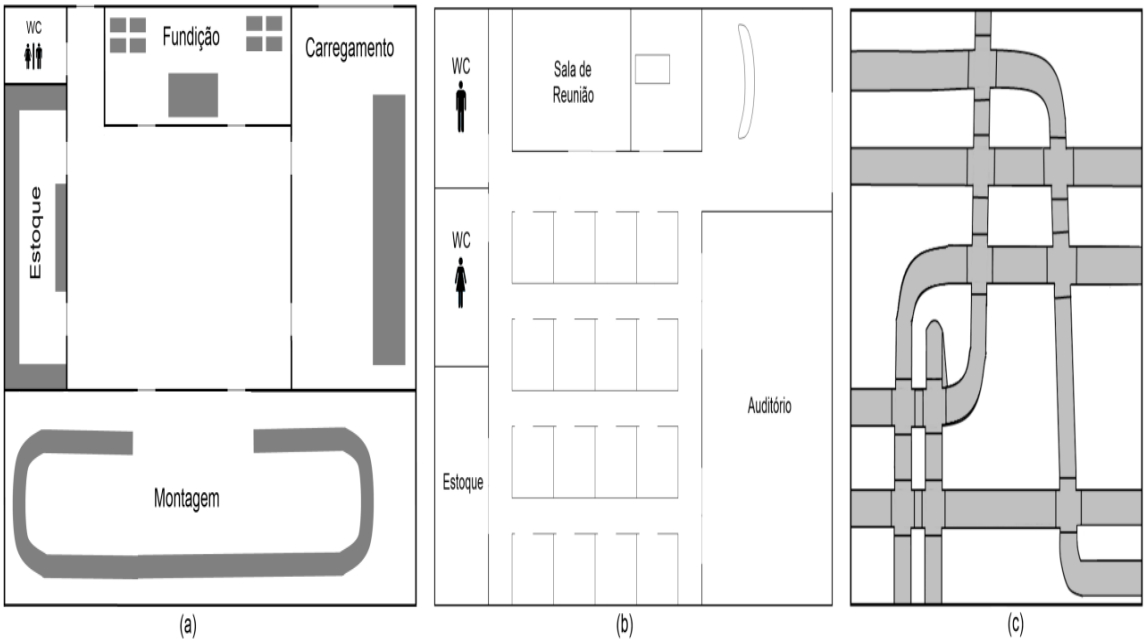


Figura 4.2 – Ambientes propostos para a simulação. (a) Fábrica; (b) Escritório; (c) Sistema de tubulação.

Na primeira hipótese o robô deve percorrer a trajetória fornecida com precisão, sem a necessidade de um planejamento de caminho. Um exemplo típico de aplicação é a solicitação que um robô realize uma operação de inspeção de um conjunto de tubos para detecção de falhas ou verificação de irregularidades.

Enquanto, na segunda hipótese, é solicitado ao robô que se desloque a um local do ambiente, fornecendo o mapa topológico e um banco de dados contendo as coordenadas da pista. Assim, o robô deverá definir o caminho através das informações disponíveis e calcular a trajetória necessária. Uma aplicação para esta hipótese seria o transporte de uma peça de um local para outro numa fábrica.

Finalmente, os resultados desta análise serão apresentados de acordo com os módulos do sistema de navegação, apontando os resultados obtidos para cada módulo separadamente. Assim, este capítulo será subdividido nos seguintes tópicos: Percepção do ambiente, Mapeamento e Localização, Planejamento do caminho, Planejamento de trajetória e Execução de trajetória.

4.2. Percepção do ambiente

A percepção de um ambiente depende incondicionalmente do sistema de sensoriamento, assim, é necessário analisar o comportamento dos sensores às possíveis entradas de forma a compreender como o robô interpreta as leituras dos sensores. Portanto, realizou-se um experimento para verificar a resposta dos transistores seguidores de linha em uma pista retilínea.

Dentro desse contexto, foi montado um experimento para encontrar a variação ΔM (diferença entre as medidas encontradas nos sensores seguidores de linha) em relação à variação da distância ΔS (distância entre a linha central do robô e a linha central da pista, ilustrada na Figura 3.19).

No experimento, o robô foi colocado numa pista retilínea de cor preta de 2cm de espessura, e efetuou-se um movimento linear do lado esquerdo para o lado direito, permitindo assim, encontrar a variação de ΔM em relação a ΔS.

O experimento foi iniciado com a linha central do robô distante 1cm à esquerda da linha central da pista, e a diferença ΔM foi anotada, em seguida, o robô foi deslocado 1mm para a direita e uma nova medida ΔM foi anotada, este procedimento se repetiu até que a linha central do robô distante 1cm à direita da linha central da pista.

Os resultados obtidos através deste experimento são apresentados na Tabela 4.1 e na Figura 4.3, onde os valores de ΔM são calculados diminuindo o valor obtido no fototransitor esquerdo pelo o valor obtido no direito. Estes valores obtidos pelos fototransistores variam de 0 a 255, sendo 0 para reflexão totalmente escura e 255 para reflexão totalmente clara.

Tabela 4.1 – Variação ΔS x ΔM.

ΔS (mm)	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
ΔM	49	48	47	44	37	35	27	19	7,2	3,7	1	-5	-8	-14	-19	-26	-34	-39	-41	-42	-43

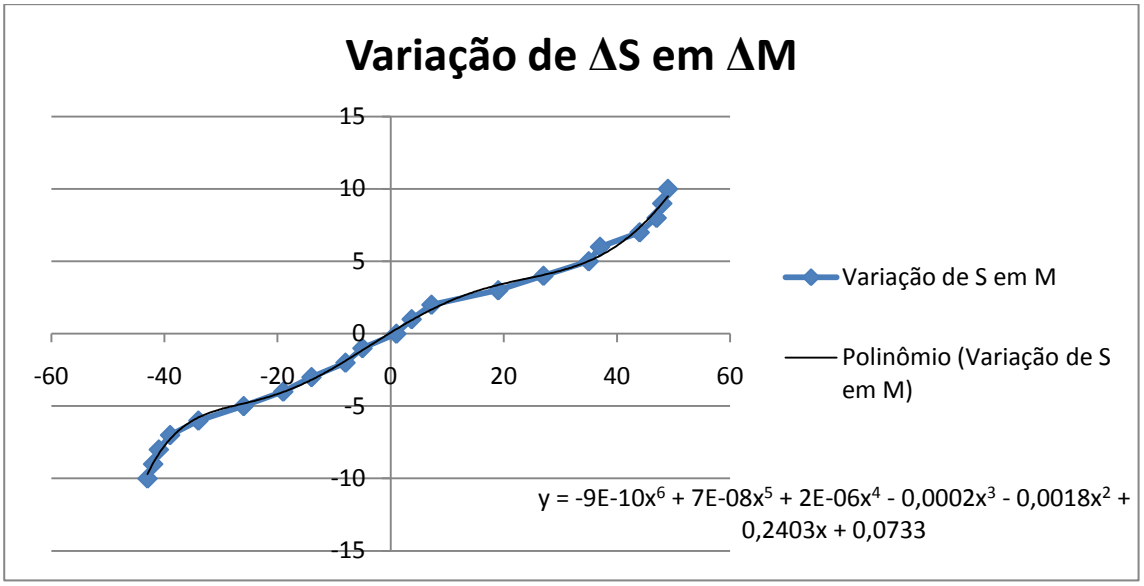


Figura 4.3 – Variação ΔSxΔM.

O polinômio calculado na Figura é a equação da linha de tendência da variação *ΔS/ΔM*, assim, esta equação pode ser usada para estimar a posição do robô em relação à linha da pista através da diferença entre os valores obtidos no transistores seguidores de linha. Caso seja necessário, ou oportuno, pode-se modificar o polinômio, reduzindo o grau do polinômio, e conseqüentemente, diminuindo o tempo de processamento.

4.3. Mapeamento e Localização

Para uma navegação precisa é importante possuir um prévio conhecimento dos mapas a serem utilizados, dessa forma, neste tópico serão mostrados detalhadamente a construção dos mapas utilizados correspondentes aos três ambientes descritos anteriormente e ilustrados na Figura 4.2.

Os mapas topológicos das pistas com os pontos de controle e de auxílio, dos três ambientes são mostrados nas Figuras 4.4, 4.5 e 4.6, onde os círculos azuis representam os pontos de caminho (nós), indicando uma posição topologicamente importantes, pelos quais o robô deve passar para se deslocar de um local a outro; os losangos amarelos representam os pontos de controle, que são utilizados para fornecer uma maior precisão na trajetória,

podendo ser alterados de acordo com a necessidade, variando o número de pontos e a sua posição; os quadrados vermelhos representam os pontos de auxílio, utilizados para auxiliar numa curva aguda, que são colocados na pista eqüidistantes ao nó, numa distância maior que *l*/2 do nó utilizado.

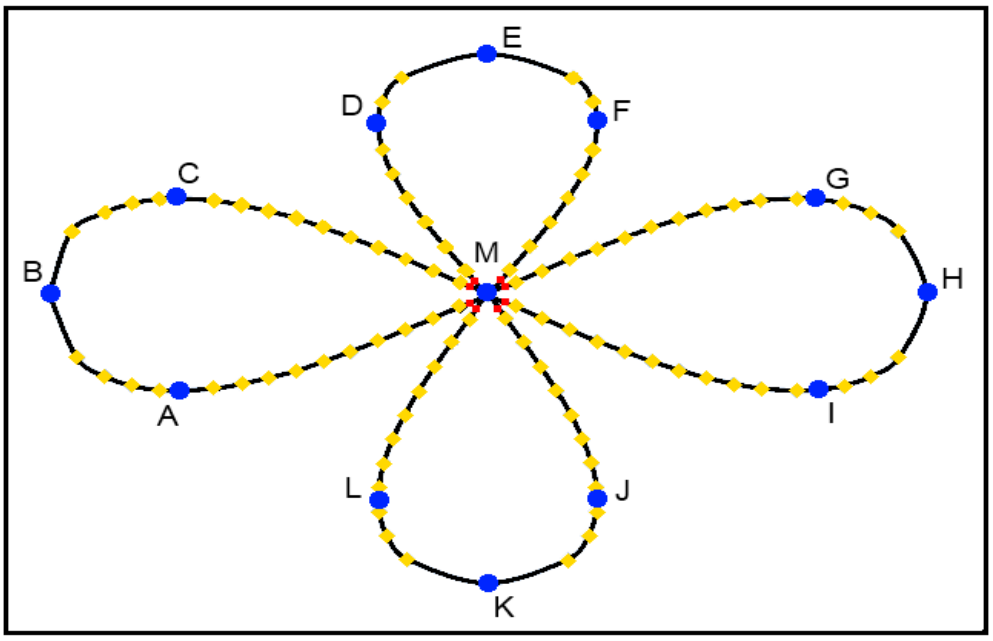


Figura 4.4 – Os mapas topológicos com os pontos de controle da fábrica.

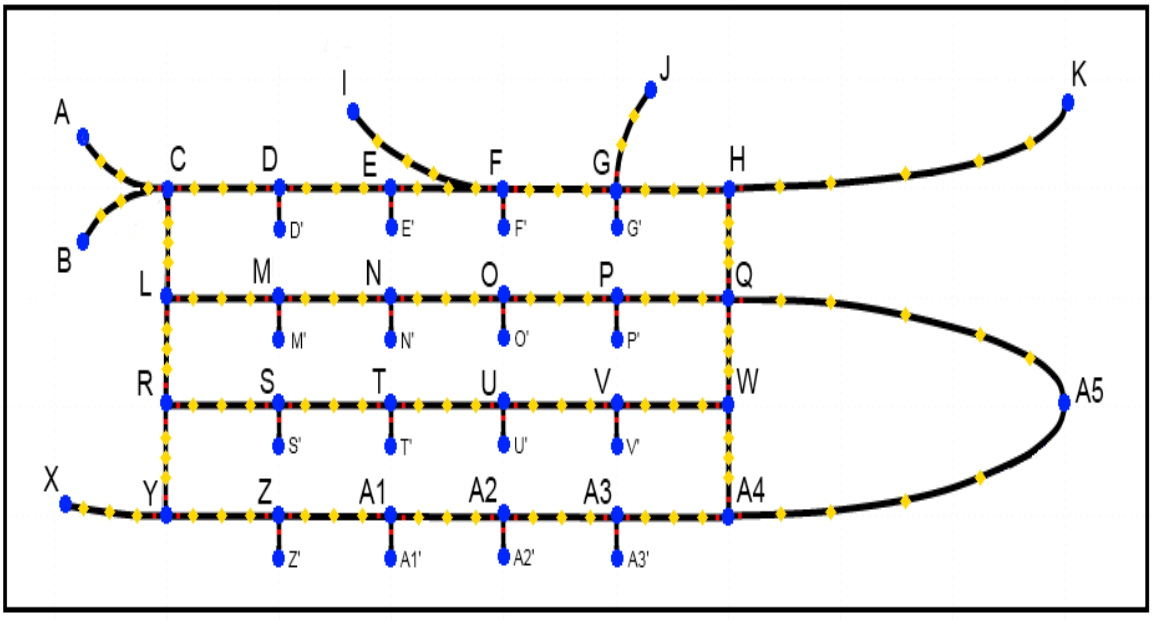


Figura 4.5 – Os mapas topológicos com os pontos de controle da escritório.

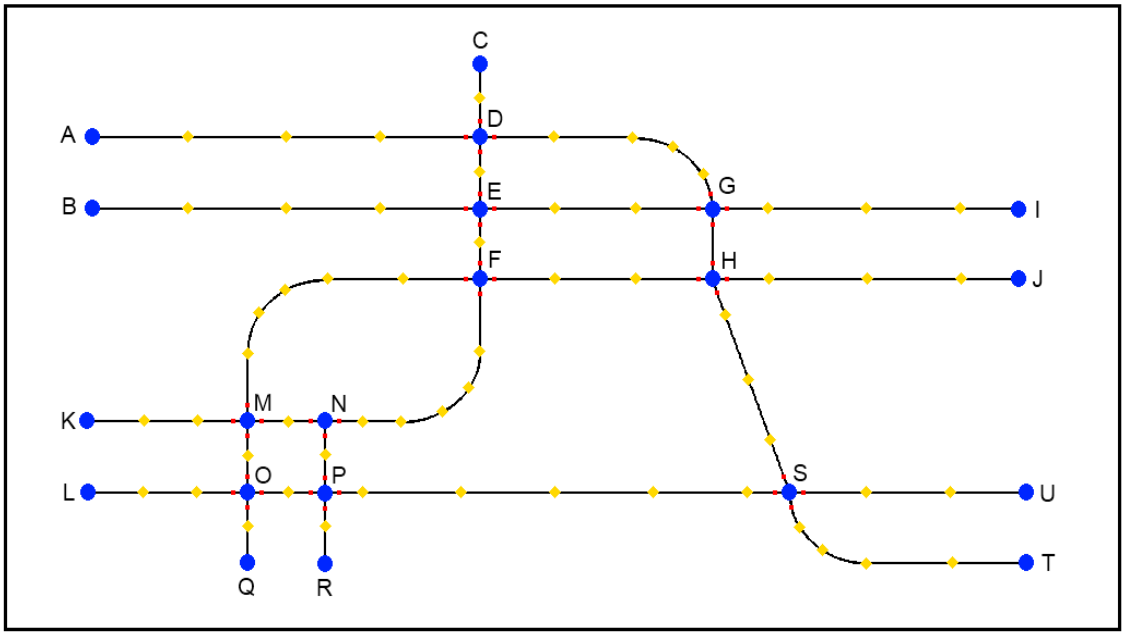


Figura 4.6 – Os mapas topológicos com os pontos de controle do sistema de tubulação.

A partir dos mapas topológicos é possível construir os grafos G = (N, E) de todos os ambientes, onde N são os nós do ambiente (pontos de caminho) e suas respectivas coordenadas, e E são os trajetos, compostos pelo nó inicial, nó final, pesos de ida e volta e a trajetória constituída pelos pontos de controle. Para exemplificar, as Tabelas 4.2, 4.3 e 4.4 mostram o grafo do ambiente da fábrica.

A definição dos pontos de auxílio será realizada de acordo com a distância entre as rodas *l*, para determinar a distância τ necessária entre o ponto de auxílio e o nó foi realizado um experimento testando três valores para τ. Devido as características dos motores do ASURO relatadas no subcapítulo 2.6, decidiu-se que as duas rodas deverão manter-se girando na curva, assim, como com τ = l/2 uma das rodas pára, os valores escolhidos para τ no experimento são *l*, 2*l* e 3*l*.

Tabela 4.2 – Coordenadas dos nós da fábrica.

N	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**
x	2,35	0	2,35	6	8	10	13,65	16	13,65	10	8	6	8
y	-2	0	-2	3,55	5	3,55	2	0	-2	-4,2	-6	-4,2	0

Tabela 4.3 – Pesos dos trajetos.

E	N_I	N_F	w_{IF}	w_{FI}	Trajetória
AB	A	B	0,92576	2,46868	T1
BC	B	C	0,92576	2,46868	T2
CM	C	M	1,19871	4,19548	T3
MA	M	A	1,19871	4,19548	T4
MF	M	F	0,81492	2,85223	T5
FE	F	E	0,7411	1,97626	T6
ED	E	D	0,7411	1,97626	T7
DM	D	M	0,81492	2,85223	T8
MG	M	G	1,19871	4,19548	T9
GH	G	H	0,92576	2,46868	T10
HI	H	I	0,92576	2,46868	T11
IM	I	M	1,19871	4,19548	T12
MJ	M	J	0,93038	3,25632	T13
JK	J	K	1,2	3,2	T14
KL	K	L	1,2	3,2	T15
LM	L	M	0,93038	3,25632	T16

Tabela 4.4 – Coordenadas dos pontos de controle das trajetórias.

Trajetória		Coordenadas					
T1	x	2,3500	2,0000	1,5000	...	0,5000	0,0000
	y	-2,0000	-1,9840	-1,8950	...	-1,3050	0,0000
T2	x	0,0000	0,4000	1,0000	...	2,0000	2,3500
	y	0,0000	1,3050	1,6940	...	1,9840	2,0000
...	...	...	...	...	...	...	...
T16	x	6,0000	6,0120	6,1050	...	7,6670	8,0000
	y	-4,2000	-4,0000	-3,5000	...	-0,5000	0,0000

A seguir são apresentados resultados comparativos dos três valores de τ para uma curva de 90º (Figura 4.7) e 60º (Figura 4.8), onde podemos observar que nenhuma das trajetórias utilizando os pontos de auxílio segue exatamente a trajetória da pista, porém, a trajetória alcançada utilizando $\tau = l$ é precisa, podendo ser utilizada conforme a tarefa desejada. Conseqüentemente, para as tarefas que exigem grande precisão do robô, torna-se

mais conveniente efetuar um movimento de rotação do mesmo no nó ao invés de utilizar os pontos de auxílio.

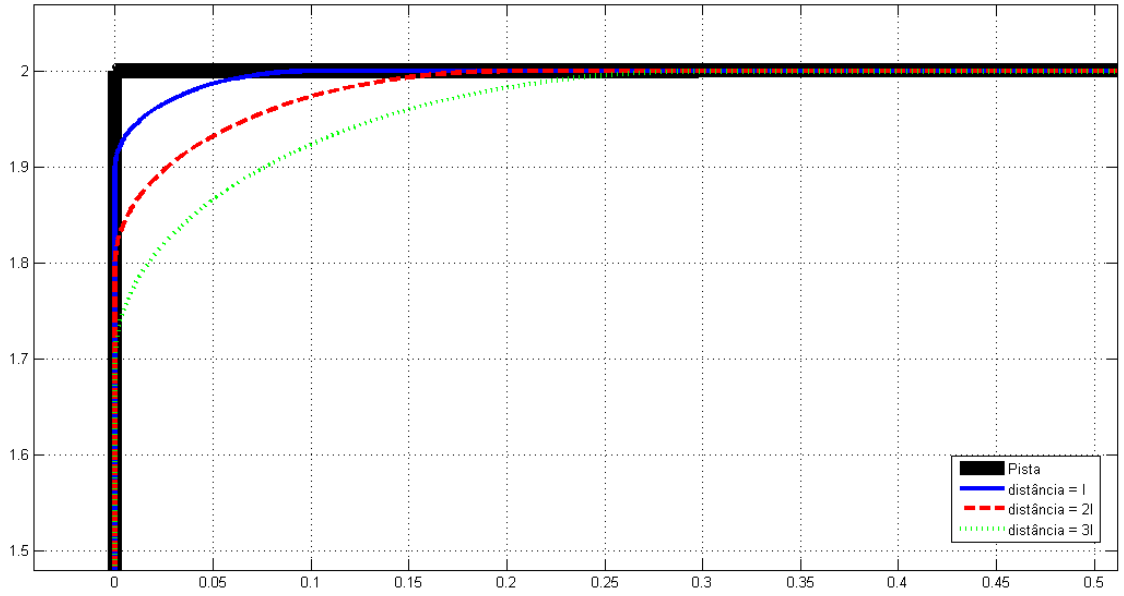


Figura 4.7 – Resultados com diferentes pontos de auxílio numa curva de 90º.

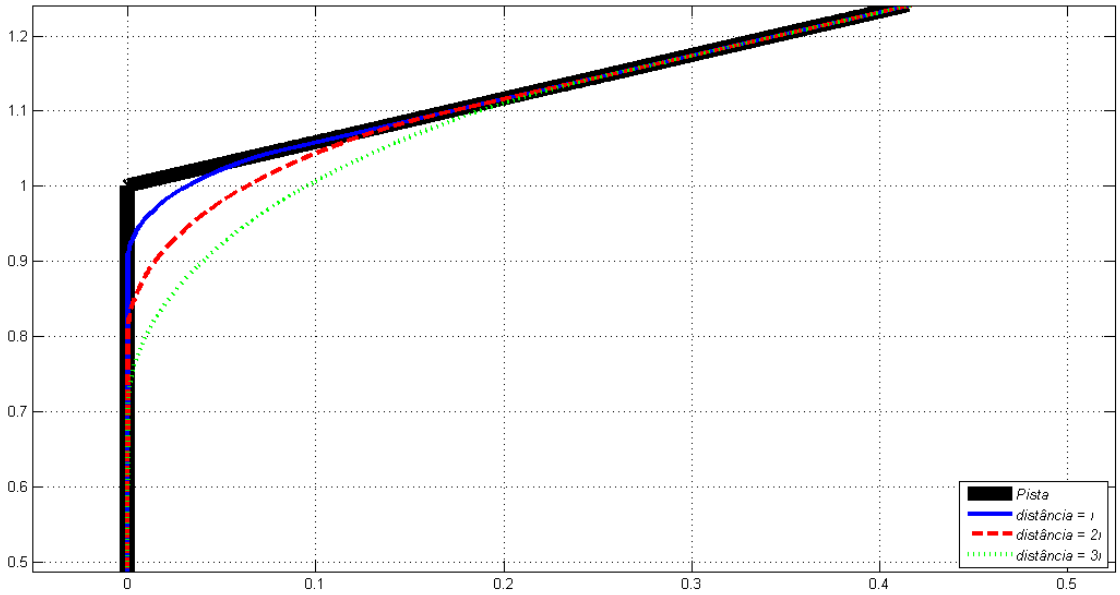


Figura 4.8 – Resultados com diferentes pontos de auxílio numa curva de 60º.

Como não necessitamos de uma precisão tão grande, utilizar-se-á os pontos de auxílio, sendo incluídos quando a curva for maior que 45º, substituindo as coordenadas dos valores do nó na trajetória desejada pelas coordenas dos dois pontos de auxílio. Os valores destes pontos são determinados para $\tau = l$. A título de exemplo, consideraremos os valores mostrados na Tabela 4.5 para os pontos de auxílio para um ambiente fabril.

Tabela 4.5 – Pontos de auxílio nas trajetórias da fábrica.

Pontos de auxílio	**$PAUX_{CM}(2)$**	**$PAUX_{MA}(1)$**	**$PAUX_{DM}(2)$**	**$PAUX_{MF}(1)$**	**$PAUX_{LM}(2)$**	**$PAUX_{MJ}(1)$**	**$PAUX_{IM}(2)$**	**$PAUX_{MG}(2)$**
x	7,9956	7,9956	7,994	8,006	7,9918	8,0081	8,046	8,0043
y	0,00902	-0,0089	0,00803	0,00795	-0,0058	-0,0058	-0,0089	0,009

Para elucidar a utilização dos pontos de auxílio, um possível exemplo seria considerar um caminho C – M – J, cujas trajetórias são CM e MJ conectadas num ângulo maior que 45º, assim, as coordenadas da trajetória seriam: CM(1), CM(2),..., CM(n-1), CM(n), MJ(1), MJ(2),..., MJ(n-1), MJ(n); onde as coordenadas do nó C seria CM(1), do nó M seria CM(n) e MJ(1), sendo CM(n) = MJ(1), e do nó MJ(n). Como o ângulo é maior que 45º, as coordenadas CM(n) e MJ(1) seriam substituídas pelos seus respectivos pontos de auxílio $PAUX_{CM}(2)$ e $PAUX_{MJ}(1)$, portanto, as coordenadas da trajetória ficariam: CM(1), CM(2),..., CM(n-1), $PAUX_{CM}(2)$, $PAUX_{MJ}(1)$, MJ(2),..., MJ(n-1), MJ(n).

4.4. Planejamento do caminho

O planejamento do caminho utiliza o modelo interno do mundo para definir os nós pelos quais o robô deve passar para se deslocar entre sua posição atual e a posição de seu objetivo. Para esta tarefa existem inúmeros algoritmos de busca, descritos detalhadamente no anexo C desta dissertação. Considerando aspectos associados a otimização e tempo de processamento neste trabalho utilizaremos o algoritmo de Dijkstra.

Considerado o mapa topológico do ambiente da tubulação (Figura 4.6), e seu grafo G = (V, E), disposto no anexo E, foram realizados alguns testes para verificar os resultados

do algoritmo de Dijkstra. A Figura 4.9 mostra os resultados obtidos para na realização dos testes envolvendo os caminhos realizados entre o nó: Q e H (Figura 4.9a), B e U (Figura 4.9b), L e J (Figura 4.9c) e H e K (Figura 4.9d).

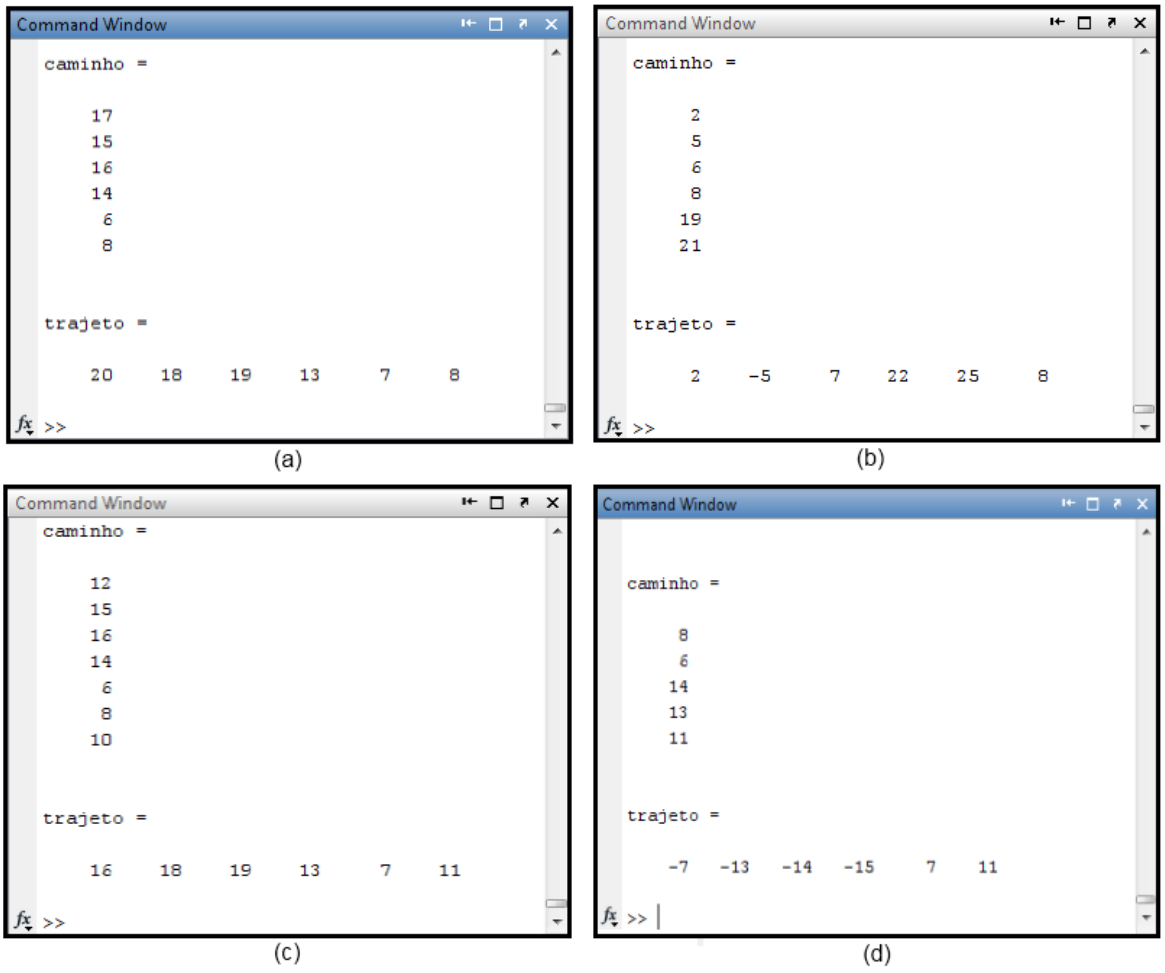


Figura 4.9 – Resultados da implementação do algoritmo Dijkstra. (a) Caminho entre os nós Q e H; (a) Caminho entre os nós B e U; (a) Caminho entre os nós L e J; (a) Caminho entre os nós H e K,

Podemos observar que no primeiro teste foi realizado o caminho entre o nó Q e H, e o caminho selecionado foi Q – O – P – N – F – H, utilizando os trajetos E20, E18, E19, E13, E7 e E8 (Figura 4.9a).

A Figura 4.9b mostra o resultado para o caminho entre o nó B e o nó U, onde a utilização da trajetória E5 é negativa, o que exprime que esta trajetória deve seguir em sentido invertido, ou seja, da última coordenada até a primeira. Exemplificando,

considerando que as coordenadas de E2 são E2(1), E2(2),..., E2(n-1) e E2(n) e de E5 são E5(1), E5(2),..., E5(n-1) e E5(n), se as trajetórias selecionadas fossem ambas positivas, a seqüência de coordenadas a ser seguida seria E2(1), E2(2),..., E2(n-1), E2(n), E5(1), E5(2),..., E5(n-1), E5(n); porém, como a trajetória E5 tem a direção negativa, a seqüência de coordenadas fica E2(1), E2(2),..., E2(n-1), E2(n), E5(n), E5(n-1),..., E5(2), E5(1).

A Figura 4.9c mostra o resultado para o caminho entre o nó L e o nó J e a Figura 4.9d mostra o resultado para o caminho entre o nó H e o nó K. Observou-se que nestes testes o algoritmo utilizou a trajetória NF, mesmo no quarto teste, onde há o trecho F – N – M, onde o natural seria utilizar a trajetória FM. Isto ocorre, pois a distância entre o nó F e o nó M utilizando o trecho F – M ou o trecho F – N – M possuem um deslocamento total semelhante, porém, o fator de segurança σ é diferente. O fator σ nas trajetórias MF e NF são os mesmos, entretanto, o fator σ da trajetória MN é menor, e esta diferença faz com que o peso total do trecho F – M seja maior que na trajetória F – N – M, inutilizando a trajetória MF.

4.5. Planejamento de trajetória

O planejamento de trajetória é responsável pelo cálculo das velocidades dos atuadores através das coordenadas dos pontos pelos quais o robô deve passar para alcançar seu objetivo. Como foi mencionando anteriormente, será utilizado uma abordagem baseada no caminho de Dubins, que permite alcançar o sub-objetivo desejado através de uma curva. Para validar esta abordagem foram realizadas algumas simulações utilizando o bloco DD&GP, para os três ambientes descritos anteriormente, utilizando os resultados obtidos pelo planejamento de caminho.

As simulações realizadas foram divididas nas categorias:

a) **Transporte:** o robô procura alcançar o seu objetivo sem afastar-se excessivamente da pista e com o menor número de pontos possível, de forma que possa realizar esta tarefa rapidamente sem se perder.

b) **Inspeção:** o robô procura seguir a pista precisamente, buscando, principalmente, permanecer o mais próximo da linha central da pista, de modo que a trajetória interfira minimamente no resultado da inspeção.

Para essas duas categorias (transporte e inspeção) foram realizadas simulações para os três ambientes descritos anteriormente. Os resultados das simulações são apresentados nas Figuras 4.10, 4.11 e 4.12 (categoria transporte) e nas Figuras 4.13, 4.14 e 4.15 (categoria inspeção).

As simulação da categoria transporte utilizaram o planejamento da caminhos, para determinar os nós pelos quais o robô deve passar, e o planejamento de trajetória para determinar as velocidades para a execução da trajetória desejada.

A Figura 4.10 mostra os resultados obtidos na simulação do ambiente de escritório, onde a Figura 4.10a mostra a simulação do deslocamento entre o nó M' até o nó B utilizando 15 pontos de controle, passando pelos nós M' – M – L – C; e a Figura 4.10b mostra a simulação do deslocamento entre o nó J até o nó A5 do escritório utilizando 20 pontos de controle, passando pelos nós J – G – H – Q – A5.

A Figura 4.11 mostra os resultados obtidos na simulação do ambiente de tubulação, onde a Figura 4.11a mostra a simulação do deslocamento entre o nó L até o nó C utilizando 30 pontos de controle, passando pelos nós L – O – M – N – F – E – D – C; e a Figura 4.11b mostra a simulação do deslocamento entre o nó I até o nó T do escritório utilizando 21 pontos de controle, passando pelos nós I – G – H – S – T.

A Figura 4.12 mostra os resultados obtidos na simulação do ambiente fabril no deslocamento entre os nós B e K (Figura 4.12a), B e H (Figura 4.12b) e B e E (Figura 4.12c) com 12, 14 e 12 pontos respectivamente. A Figura 4.12d mostra o resultado da simulação do deslocamento por todos os nós do ambiente fabril utilizando 53 pontos de controle, passando pelos nós na seqüência B – C – M – J – K – L – M – G – H – I – M – D

–E – F – M – A – B. Essa última trajetória foi escolhida devido sua complexidade de execução devido a constante mudança de direção.

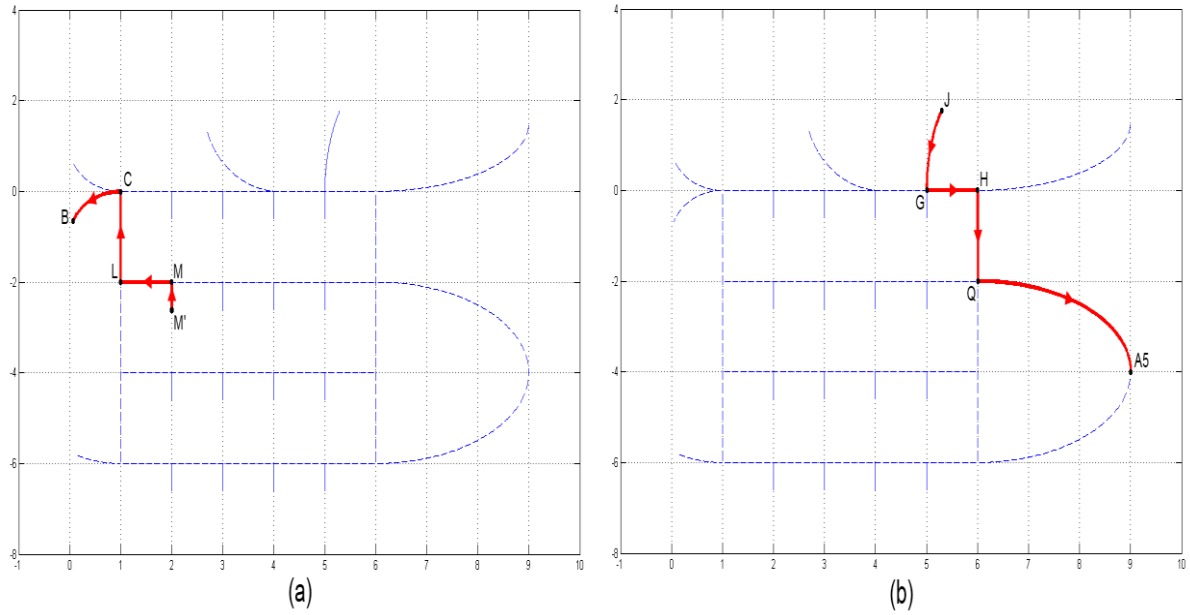


Figura 4.10 – Simulação de transporte no escritório. (a) Simulação do caminho entre os nós M' e B; (b) Simulação do caminho entre os nós J e A5.

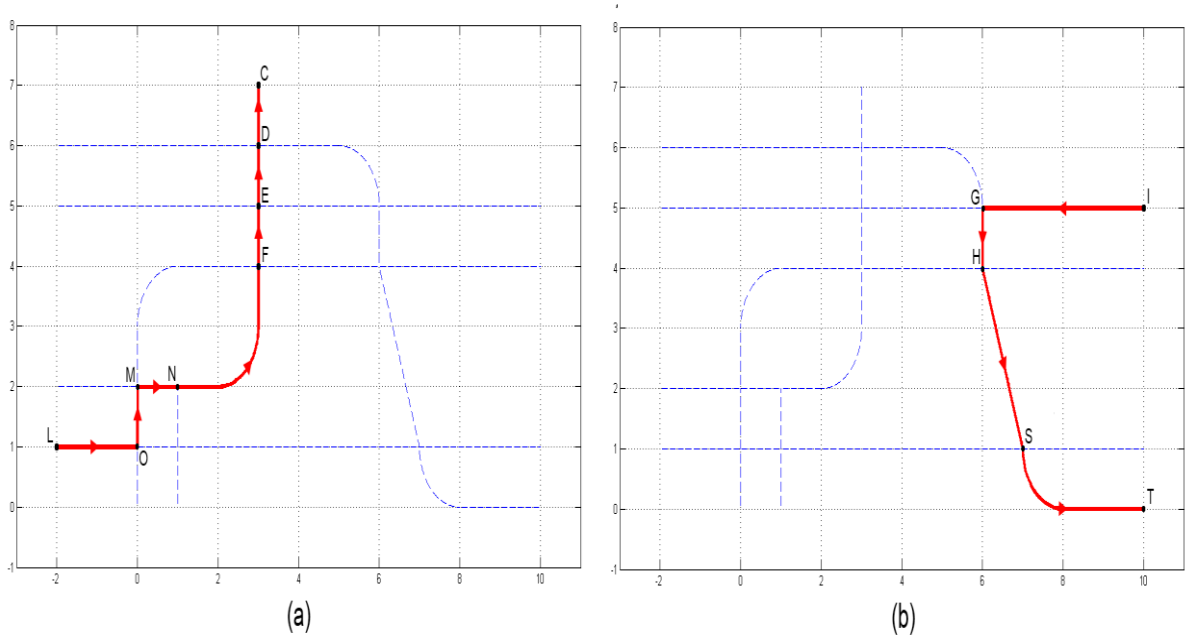


Figura 4.11 – Simulação de transporte no sistema de tubulação. (a) Simulação do caminho entre os nós L e C; (b) Simulação do caminho entre os nós I e T.

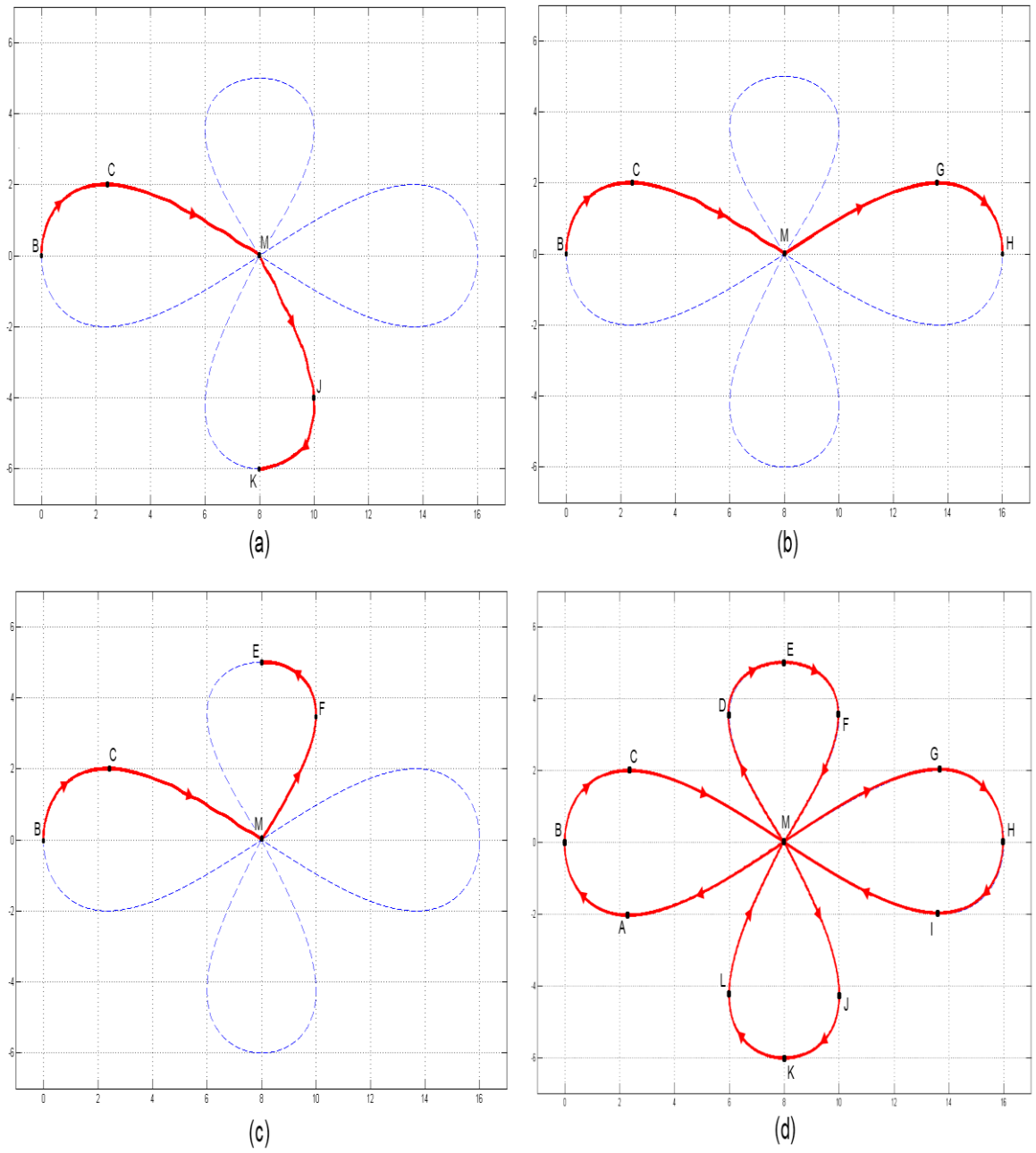


Figura 4.12 – Simulação de transporte na fábrica. (a) Simulação do caminho entre os nós B e K; (b) Simulação do caminho entre os nós B e H; (c) Simulação do caminho entre os nós B e E; (d) Simulação do caminho passando por todos os nós saindo do nó B.

Nas simulações realizadas na categoria transporte os resultados demonstraram que apesar do uso de poucos pontos, a trajetória planejada possui precisão suficiente para seguir a linha, sendo possível sua utilização na tarefa de transporte ou deslocamento de materiais.

Nas simulações da categoria inspeção utilizaram os pontos definidos pelo operador para calcular os movimentos necessários para efetuar a trajetória desejada com precisão. Os resultados destas simulações são mostrados nas Figuras 4.13, 4.14 e 4.15, que utilizam um número de pontos maior do que para as simulações da categoria transporte, aproximadamente 1700, 1200 e 3100 pontos, aumentando o tempo de execução do percurso. Contudo, como observado nas Figuras 4.13, 4.14 e 4.15, o robô se deslocou com o mínimo desvio, seguindo a trajetória desejada de forma precisa, demonstrando sua possível utilização em qualquer tarefa que necessite uma alta precisão, como a inspeção.

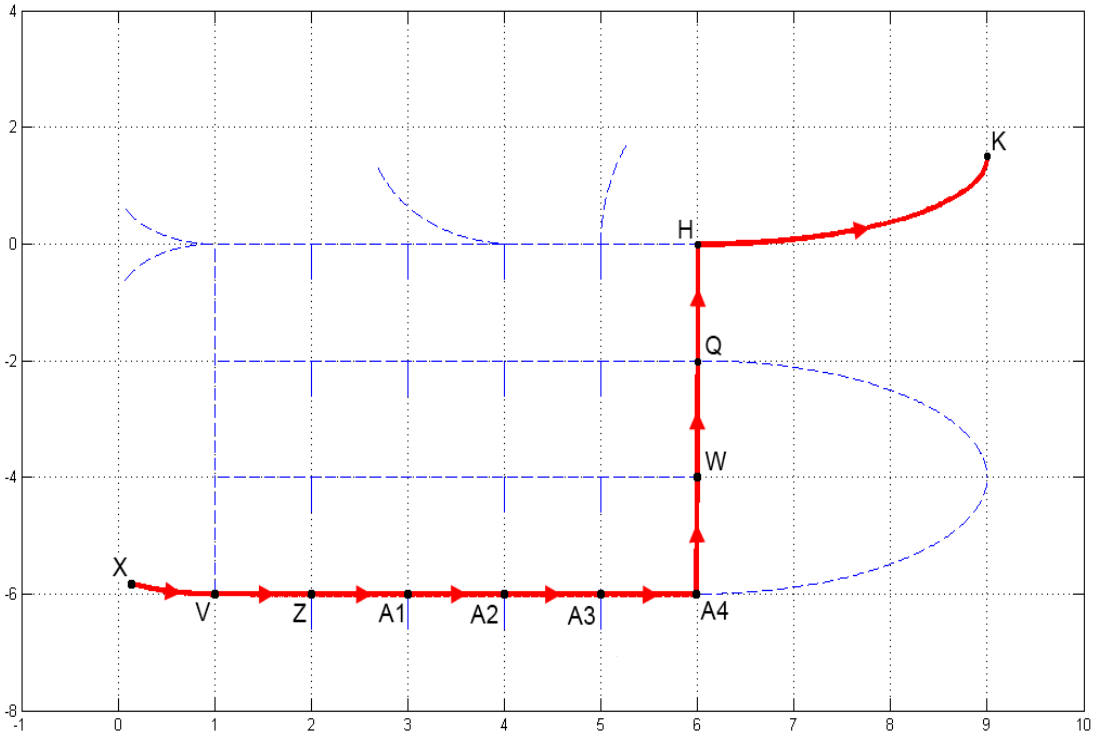


Figura 4.13 - Simulação de inspeção no escritório.

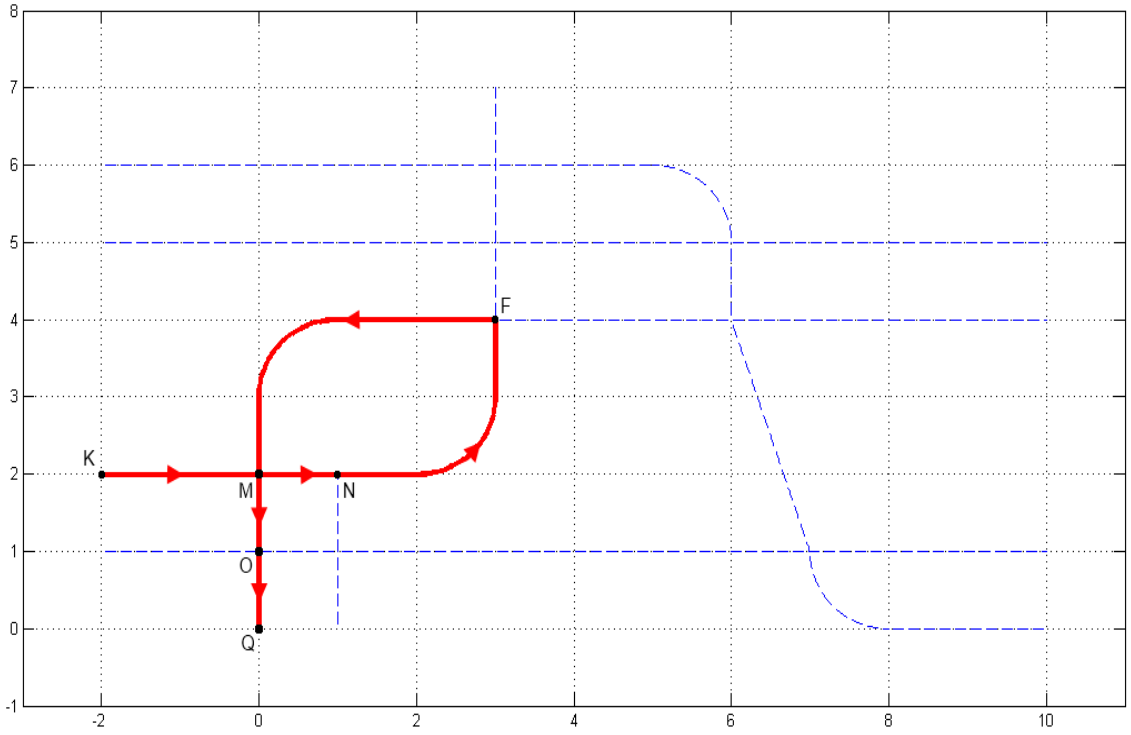


Figura 4.14 – Simulação de inspeção no sistema de tubulação.

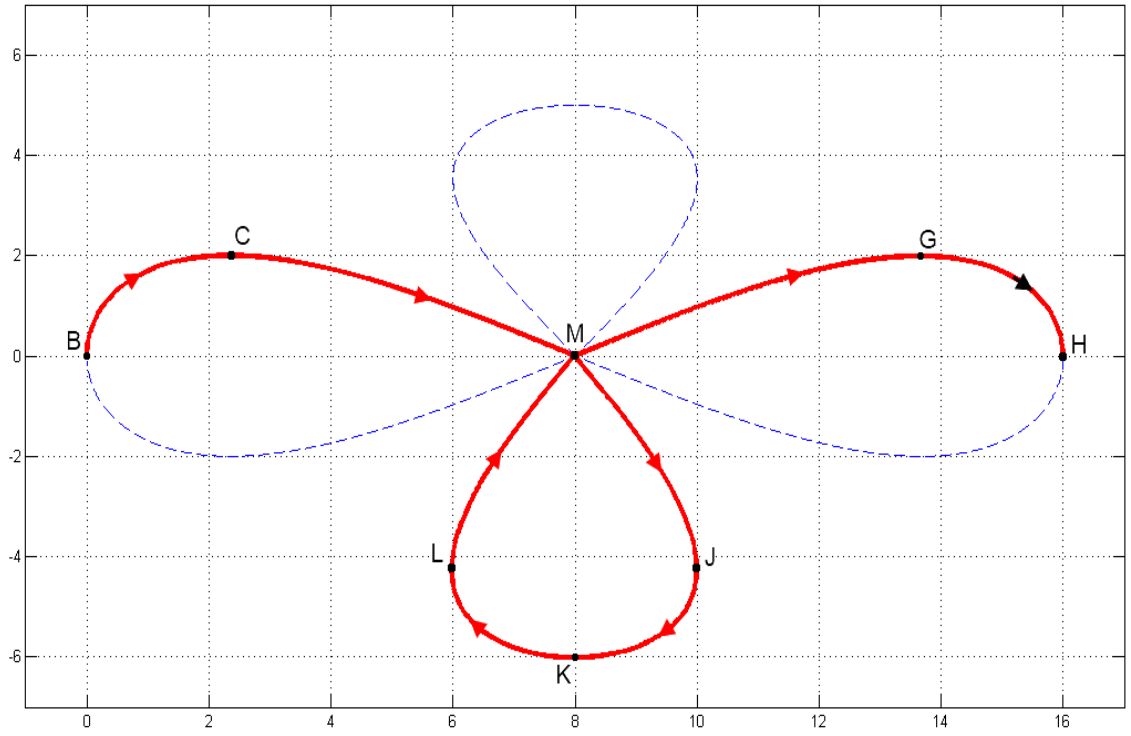


Figura 4.15 – Simulação de inspeção na fábrica.

4.6. Execução da trajetória

O módulo de execução da trajetória baseia-se no motor–schemas, realizado através do cálculo de um vetor final obtido a partir da soma dos vetores resultantes de cada comportamento desenvolvido no robô utilizando as equações (3.17), (3.18) e (3.19). Para isto é necessário calcular as velocidades para cada comportamento e estipular os pesos para cada estado: caminho livre, caminho obstruído e robô perdido.

Nos estados "caminho obstruído" e "robô perdido" os comportamentos ativos serão Evitar obstáculos e Alerta, como mostrado na Tabela 3.1, que deverão parar o robô e sinalizar, de modo a evitar maiores danos.

No estado "caminho livre", os comportamentos ativos serão Seguir o objetivo e Seguir linha, sendo o primeiro relativo à parte deliberativa e o segundo à parte reativa, portanto as velocidades resultantes do comportamento deliberativo serão as velocidades calculadas pelo planejamento de trajetória e as velocidades resultantes do comportamento reativo serão as velocidades calculadas através das equações (3.15) e (3.16) utilizando a percepção do ambiente. Os pesos P_d e P_r, mostrados na Tabela 3.1, serão definidos através de simulações ilustradas a seguir, que considerarão a resposta do robô em três trajetos distintos: reta, curva e misto.

As velocidades do comportamento deliberativo são encontradas através do planejamento de trajetória. Para as velocidades do comportamento reativo foi necessário incluir dois blocos no simulador, um para determinar a distância entre robô e a trajetória desejada (CDRT) e outro para determinar as velocidades (Velocidades Reativas), além disso, foram incluídos blocos para geração de números randômicos para simular erros no solo, no motor, etc. A Figura 4.16 mostra como ficou o simulador após as mudanças.

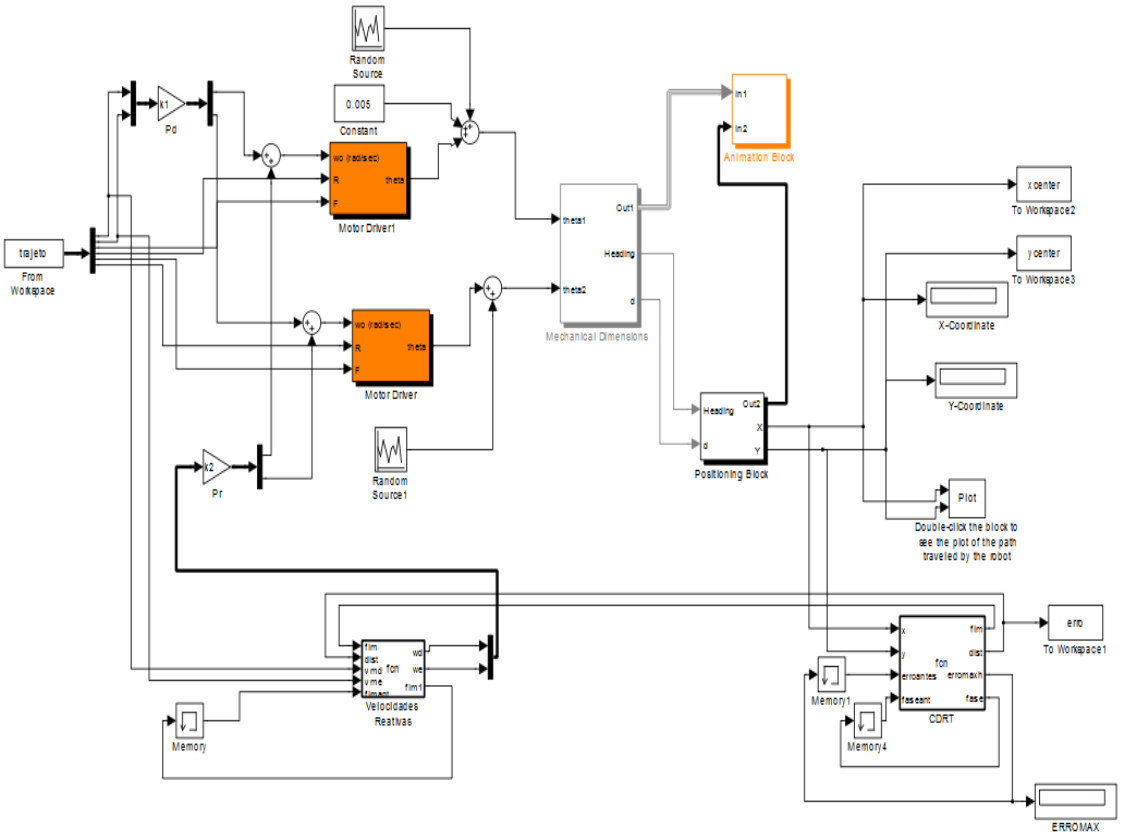


Figura 4.16 – Montagem dos blocos de simulação após acoplamento da parte reativa.

O bloco Velocidades Reativas foi implementado a partir das equações (3.15) e (3.16), considerando que V_M varia de 0 a 0,3 m/s dependendo do valor calculado no planejamento de trajetória, ΔS é determinado pelo módulo CDRT e *largp* é 2 cm, contudo, é necessário estabelecer K_R, para isso, antes da simulação de cada trajeto, foram realizadas simulações para verificar a eficiência de alguns K_R's (1 a 15) no trajeto desejado utilizando apenas o componente reativo.

A primeira trajetória simulada consiste de uma reta de 5 m (Figura 4.17a), e a segunda de uma curva de raio 2 m (Figura 4.17b), ambas as trajetórias foram simuladas considerando falhas leves e falhas graves, incluídas propositadamente para simular erros do motor, erro de posicionamento, defeitos na pista, derrapagens, diferença entre as rodas, etc. As falhas leves, mostradas em vermelho na Figura 4.18a, foram simuladas através da soma de números randômicos entre -0.001 e 0,001 às saídas dos motores; e as falhas graves,

mostradas em vermelho na Figura 4.18a, foram simuladas através da soma de números randômicos entre -0,01 e 0,01 às saídas dos motores e a adição de 0,005 à saída de um dos motores, referente à diferença entre o tamanho das rodas. Além disso, para efeito demonstrativo, nas ilustrações da Figura 4.18, é mostrada em azul as velocidade máxima dos motores.

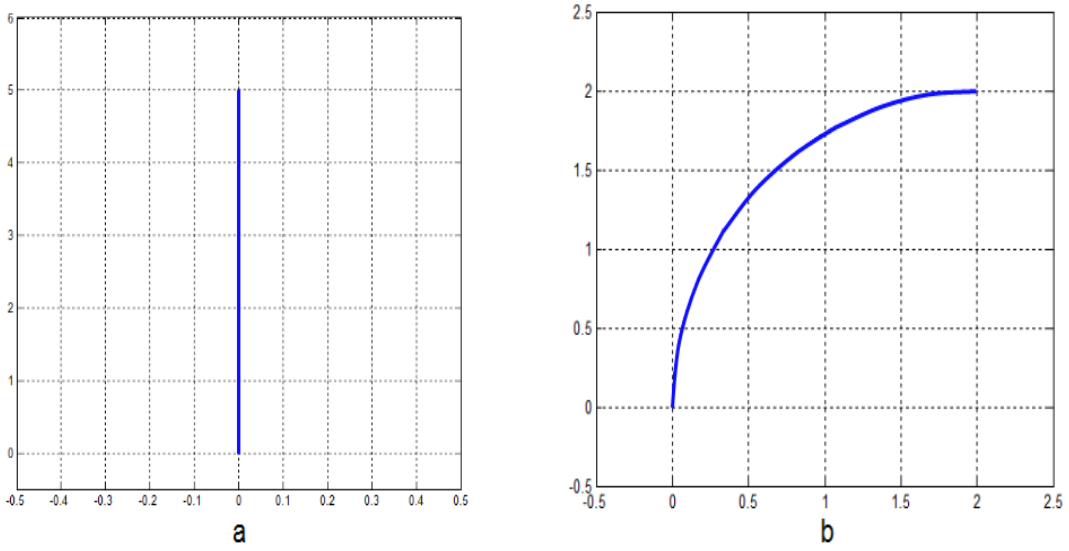


Figura 4.17 – Trajetórias simuladas. (a) Reta; (b) Curva.

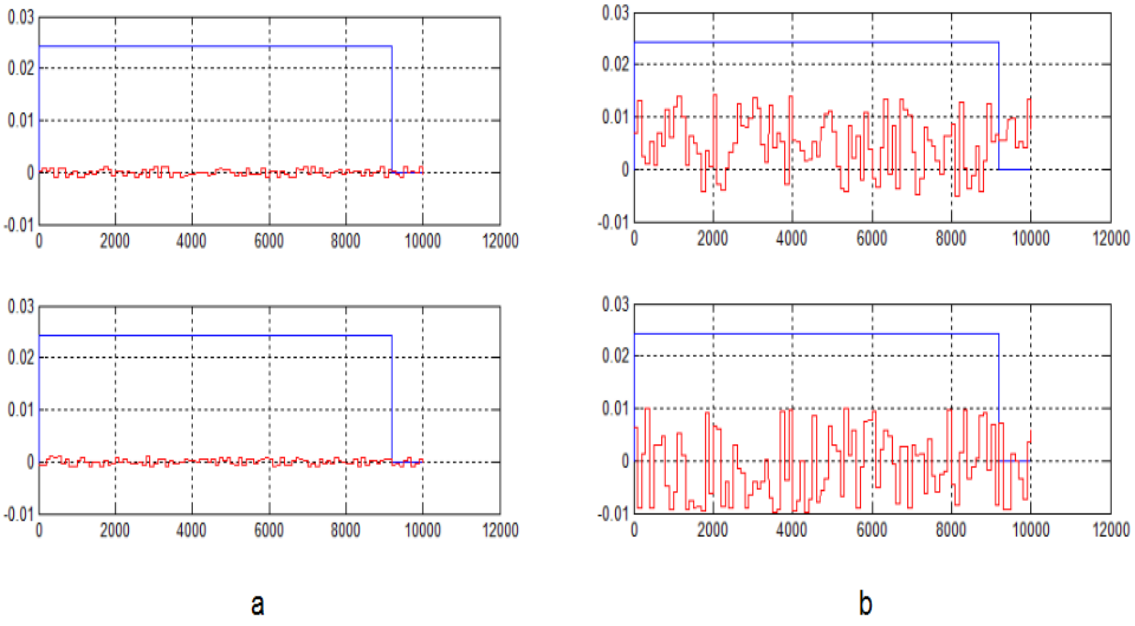


Figura 4.18 – Falhas randômicas inseridas nos motores. (a) Falha leve; (b) Falha grave.

Para comparar os resultados obtidos e definir os valores de P_d, P_r e K_R serão utilizados o erro quadrático e o erro máximo encontrados na simulação dos trajetos, sendo a equação do erro quadrático calculado pela equação (4.1), onde p_i é a posição atual e p_{di} é a posição desejada.

$$\Delta x = \sqrt{\frac{\sum_{i=1}^{n} (p_i - p_{di})^2}{n * (n-1)}} \tag{4.1}$$

Inicialmente, foram definidos os valores de K_R para cada simulação através da análise da execução das trajetórias utilizando o erro quadrático da execução para cada valor de K_R (Figuras 4.19a e 4.20a), e através do erro máximo encontrado em cada simulação (Figuras 4.19b e 4.20b).

As simulações realizadas com a primeira trajetória são mostradas na Figura 4.19, e com a segunda trajetória na Figura 4.20, onde a curva tracejada em azul são os valores do erro de K_R com a simulação com falha leve, e a curva em vermelho com falha grave.

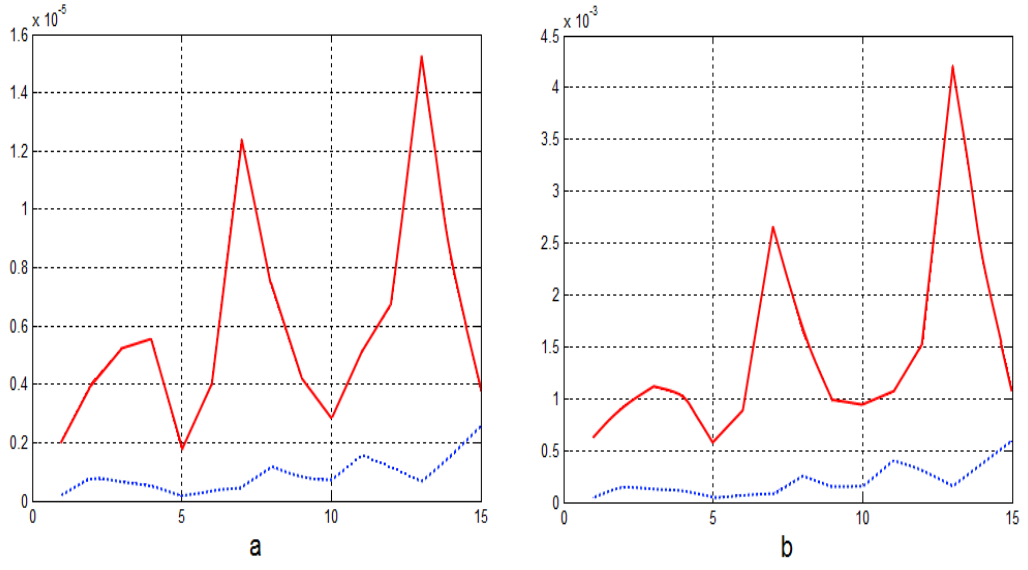


Figura 4.19 – Erros obtidos com falhas leves na simulação dos KR's. (a) Erro quadrático; (b) Erro máximo.

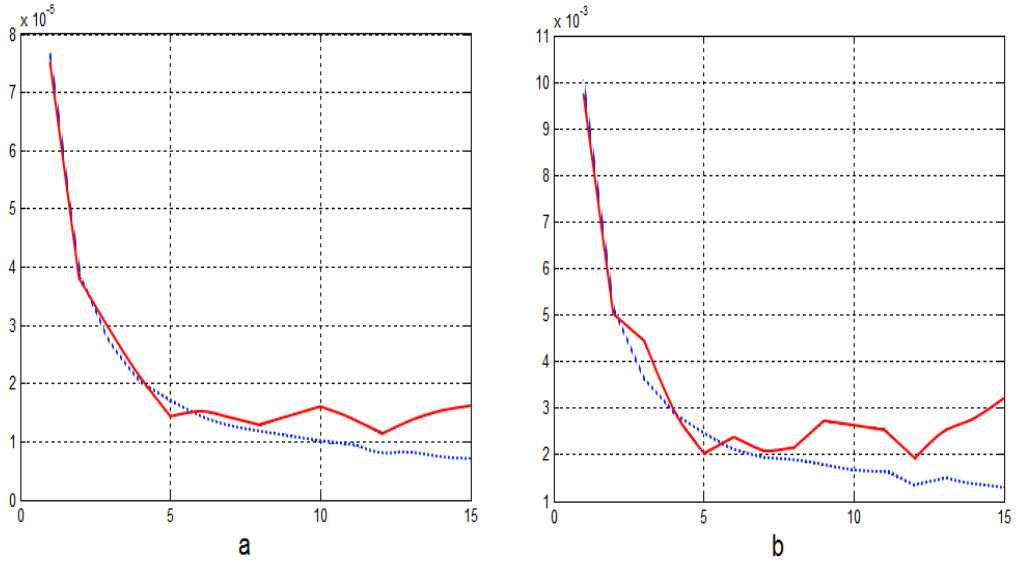


Figura 4.20 – Erros obtidos com falhas graves na simulação dos KR's. (a) Erro quadrático; (b) Erro máximo.

Como observado nas Figuras 4.19 e 4.20, o KR com o menor erro encontrado para na reta é 5, enquanto, para na curva é 12, dessa forma, esses serão os valores utilizados para cada uma das simulações concernentes.

Posteriormente, para determinar os pesos P_d e P_r ideais, foram realizadas as simulações da reta e da curva variando cada peso de 0 a 1, de forma que a soma dos pesos seja igual a 1, indicando que a velocidade total não deverá ultrapassar as velocidades máximas desejadas. A Figuras 4.21 e 4.22 mostram os resultados destas simulações das trajetórias considerando falhas leves e graves, onde o valor 0 do eixo X representa o par (0;1) para os valores de (P_d, P_r), representando uma navegação puramente reativa; o valor 11 representa o par (1;0), representando uma navegação puramente deliberativo; e os demais valores (2; 3; 4;...;10) representam uma navegação híbrida, aumentando o valor de P_d (0,1; 0,2; 0,3; ...;0,9) e diminuindo o valor de P_r (0,9; 0,8; 0,7; ...;0,1) de acordo com a variação em X. Além disso, é necessário ressaltar a importância da análise do gráfico do erro máximo, pois, a partir dele podemos averiguar se o robô manteve-se na pista ou não, onde, se o erro ultrapassar 0,01m significa que o robô saiu da pista e provavelmente se perderá.

Os resultados das simulações das trajetórias considerando falhas leves são mostrados nas Figuras 4.21a (erro quadrático) e 4.21b (erro máximo), onde as curvas tracejadas em azul são os resultados da simulação da reta e em vermelha da trajetória em curva. A partir deles é possível observar que na reta o melhor resultado foi obtido com o par puramente reativo (0;1), enquanto na curva o par puramente deliberativo (1;0) obteve um resultado expressivamente melhor, contudo, é possível observar que o par (0,8;0,2) obteve um excelente resultado na reta e um bom resultado na curva, sendo o segundo melhor resultado em ambas simulações.

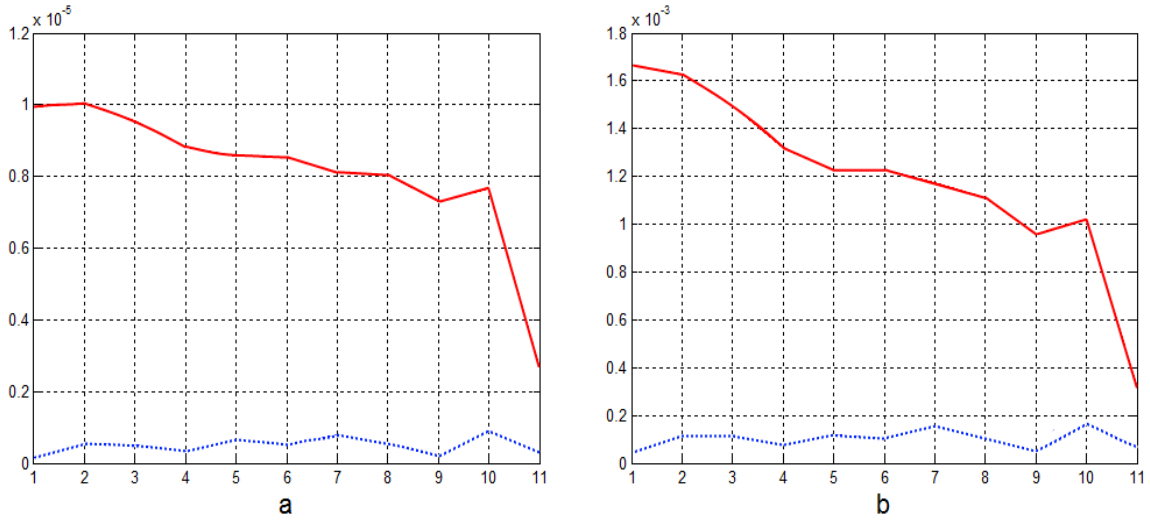


Figura 4.21 – Erros obtidos com falhas leves na simulação das trajetórias. (a) Erro quadrático; (b) Erro máximo.

Os resultados das simulações das trajetórias considerando falhas graves são mostrados nas Figuras 4.22a e 4.22b, dispostas da mesma forma que anteriormente. Observa-se que nestes resultados o par puramente deliberativo obteve resultados ruins, bem inferiores aos demais, inclusive chegando próximo a beira da pista. Nestes resultados, os pares que obtiveram melhor resultados foram (0;1), (0,5;0,5), (0,3;0,7) e (0,2;0,8) na reta, e (0,9;0,1), (0,6;0,4) e (0,8;0,2) na curva, e assim, os melhores resultados não foram obtidos pelos mesmos pares nesta simulação, complicando a definição dos pesos.

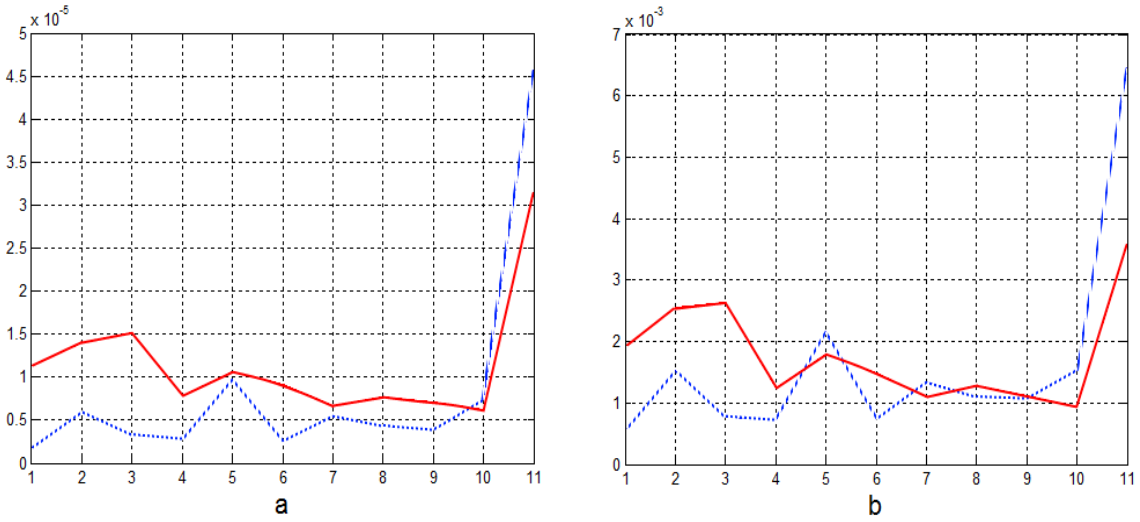


Figura 4.22 – Erros obtidos com falhas graves na simulação das trajetórias. (a) Erro quadrático; (b) Erro máximo.

O par (0,8; 0,2) foi o único que repetiu um bom resultado em mais de um experimento, mas não o fez em todos, assim, para analisar todos os resultados resolveu-se somar todas as curvas obtidas e verificar qual dos pesos obtiveram os menores erros considerando todas as simulações. O gráfico obtido é mostrado na Figura 4.23, onde a linha em verde é o resultado da soma dos erros das simulações, pelo qual se pode verificar que os pares (0,8; 0,2), (0,3; 0,7), (0,5;0,5) e (0,9;0,1) foram os resultados que se sobressaíram, mostrando que seus resultados são bons se todos as simulações forem consideradas.

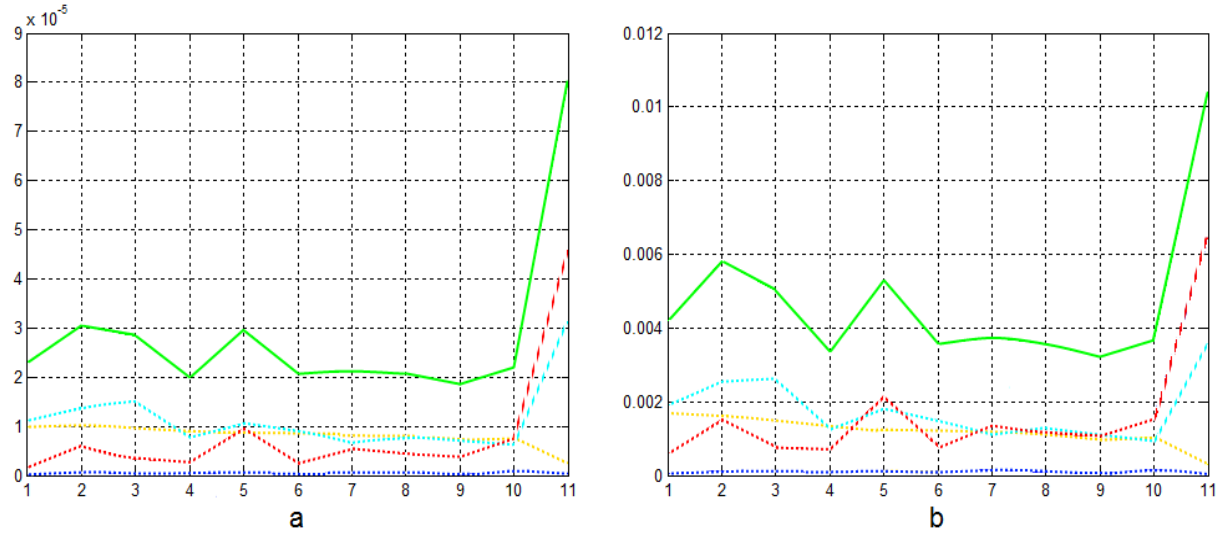


Figura 4.23 – Soma dos erros obtidos nas simulações. (a) Erro quadrático; (b) Erro máximo.

Por fim, para definir os pesos, foi realizada uma última simulação, onde buscou-se seguir uma trajetória composta por retas e curvas, mostrada na Figura 4.24, com a inserção de falhas graves. Para este o K_R selecionado foi o valor 10, pois ao analisar as quatro simulações para os K_R's observou-se que o valor 10 obteve bons resultados em todas, enquanto os valores 5 e 12 mostraram-se excelentes em duas simulações, mas precários nas outras.

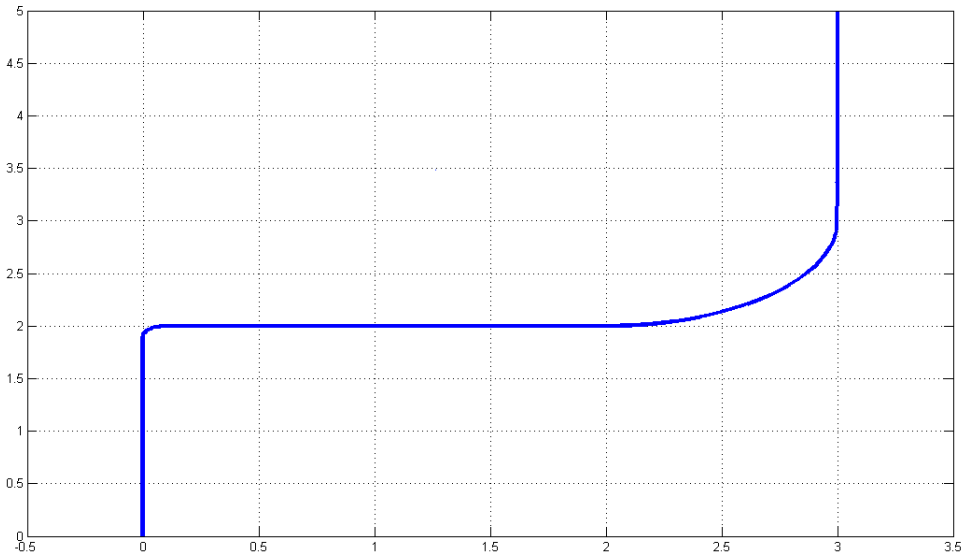


Figura 4.24 – Trajetória composta simulada.

Da mesma maneira que as simulações anteriores, a trajetória foi simulada para os 11 pares de pesos obtendo os resultados da Figura 4.25, pelo qual observou-se que os pares (0,8;0,2) e (0,9;0,1) foram os únicos que não obtiveram erro máximo maior que 0,01, e assim, os únicos que garantem sua permanência na pista mesmo com falhas graves.Portanto, devido aos bons resultados do par (0,8;0,2) nas simulações, como mostrado na Figura 4.26, este foi selecionado como sendo os pesos do sistema.

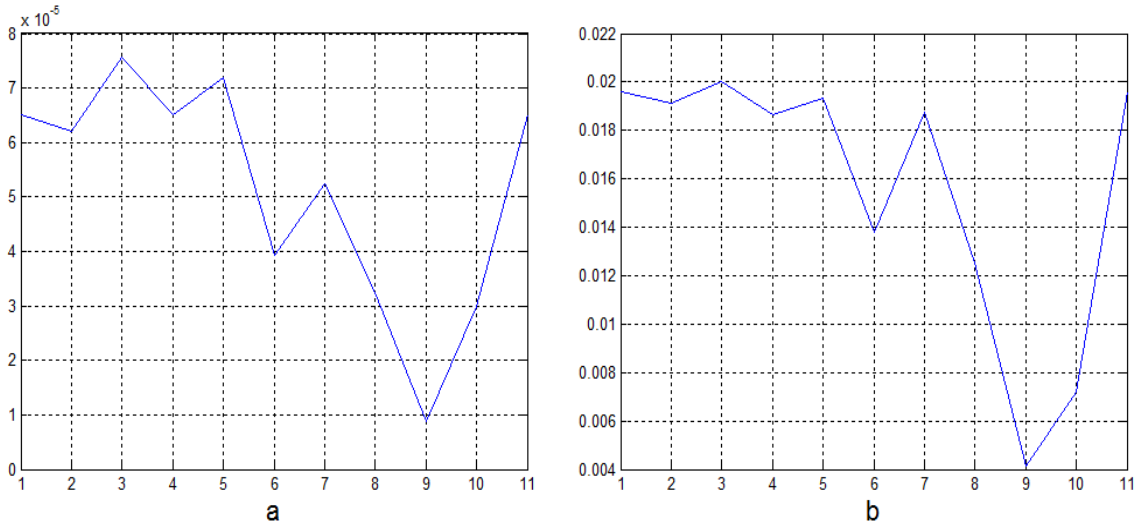


Figura 4.25 – Erros obtidos na simulação da trajetória composta. (a) Erro quadrático; (b) Erro máximo.

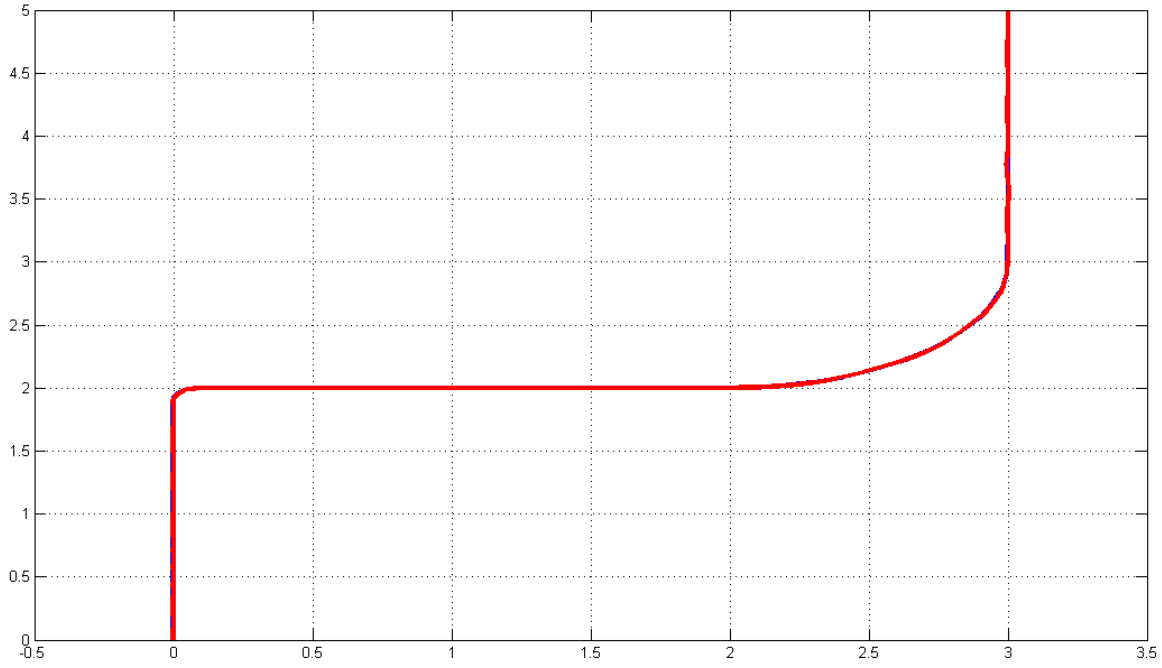


Figura 4.26 – Trajetória obtida na simulação da pista composta utilizando os pesos (0,8;0,2).

Portanto, a tabela com os pesos finais dos comportamentos, a ser utilizada pelo controlador de esquemas, ficou como mostrado na Tabela 4.6.

Tabela 4.6 – Pesos finais dos comportamentos em cada estado.

	Caminho livre	Caminho obstruído	Robô perdido
P1	0,8	0	0
P2	0,2	0	0
P3	0	1	1
P4	0	1	1

Para validar a seleção dos pesos, foi realizado um experimento (mostrado na Figura 4.27) com o robô ASURO numa pista retilínea de 2cm no chão. Neste experimento o robô manteve-se na pista como desejado nos diversos testes realizados utilizando um planejamento de trajetória pré-realizado, K_R igual a10 e os pesos mostrados na Tabela 4.6.

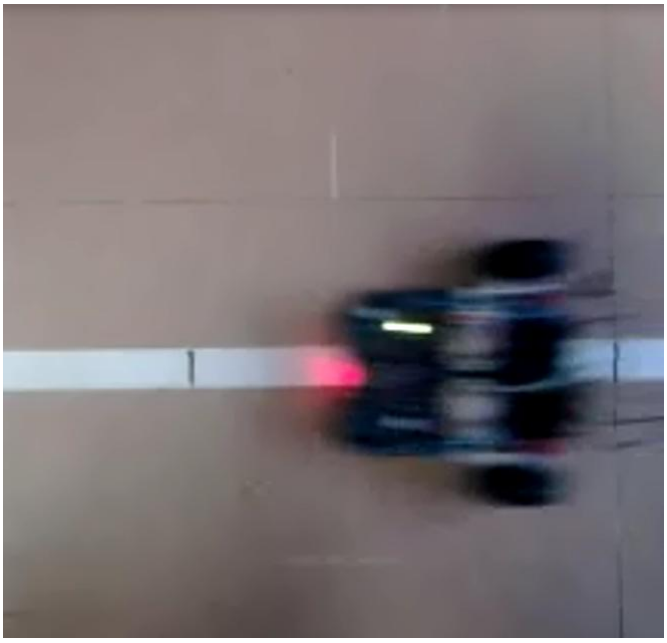


Figura 4.27 – Experimento com o robô ASURO.

4.7. Conclusões

Neste capítulo foram apresentados os principais resultados obtidos em ambientes simulados com o toolbox DD&GP e em experimentos utilizando o robô ASURO, demonstrando assim a viabilidade da utilização do sistema de navegação proposto nesta dissertação de Mestrado, onde todos os modelos implementados foram baseado na

plataforma robótica móvel ASURO, disponível no Laboratório de Automação Integrada e Robótica da Faculdade de Engenharia Mecânica da UNICAMP.

CAPÍTULO 5

Conclusões e Perspectivas Futuras

Dispositivos robóticos móveis possuem a capacidade de interagir em diversos ambientes, o que torna-os extremamente útil para a realização de diversas tarefas como no transporte numa indústria, na inspeção de um sistema de tubulação ou na limpeza de um ambiente. Porém, os sistemas de navegação existentes são limitados e não apresentam um comportamento autônomo, tornando-os incapazes de serem utilizados no mundo real (HEINEN, 2002), e por isso, este trabalho visa contribuir para o desenvolvimento de uma navegação robusta e confiável.

Desta forma, nesta dissertação de mestrado é apresentado um sistema de navegação autônoma para dispositivos robóticos móveis capaz de operar e de se adaptar a diferentes ambientes e condições, com ênfase no planejamento de trajetória. Para isso, foi necessário analisar as arquiteturas de controle de robôs móveis, selecionando a mais adequada; estudar as técnicas pra navegação robótica; propor um sistema de navegação; e validar as abordagens selecionadas através de simulações no ambiente MATLAB-Simulink® e experimentos com o robô ASURO.

O sistema de navegação proposto é baseado na arquitetura híbrida AuRA, que organiza a navegação em 5 módulos: Planejador da Missão, Raciocínio Espacial, Seqüenciador do Plano, Representação do Ambiente e Controlador de Schemas. Assim, o Planejador da Missão é o componente de interface com os usuários, sendo utilizado para receber as informações do usuário, como mapa do ambiente, ponto de controle, objetivos. Para o Raciocínio Espacial foi utilizado um planejador de caminhos, que ao receber a localização do objetivo, define os locais pelo qual o robô deverá passar para atingir seu objetivo. Para o Seqüenciador do Plano foi utilizado uma abordagem baseada no caminho de Dubins que determina os movimentos necessários para se deslocar no ambiente até

atingir seu objetivo. A Representação do ambiente é realizada através de um mapa topológico das pistas que o robô deve seguir para deslocar-se no ambiente. E, o Controlador de Schemas é responsável pela mudança dos pesos de cada comportamento de acordo com a situação atual do robô, de forma a intervir no resultado final da soma vetorial de todos os comportamentos.

O planejamento de caminho foi implementado através do algoritmo de Dijkstra que utiliza os pesos de cada trajeto E_{ij} do grafo G=(N, E) para determinar por quais nós de N o robô deve passar para atingir seu objetivo. Este algoritmo foi validado através de simulações utilizando o MATLAB®, obtendo resultados, mostrados no capítulo 4 deste trabalho, eficazes e de forma rápida, demonstrando sua otimalidade, entretanto, este algoritmo possui a limitação de encontrar o melhor caminho apenas em casos de um único objetivo, não podendo ser utilizada em casos multi-objetivos.

O Planejamento de trajetória baseado no caminho de Dubins consiste na utilização de curvas para se deslocar entre os pontos de controle, sendo muito eficaz para robôs com direção diferencial. Os resultados obtidos mostraram que esta técnica é muito eficiente, obtendo um excelente resultado com poucos pontos, e assim, utilizando um baixo custo computacional, além disso, pode-se aumentar a precisão aumentando o número de pontos de controle. Contudo, a escolha da localização dos pontos pode ser determinante para o sucesso do planejamento, sendo interessante colocar os pontos de controle nas extremidades das curvas e separá-las eqüidistantemente. Outra restrição desta técnica é a trajetória alcançada em curvas obtusas, porém, a utilização de pontos de auxílio mostrou-se eficiente, limitando esta restrição em pouquíssimos casos onde há diversos pontos próximos ligados por curvas obtusas consecutivas.

Os dois principais comportamentos desenvolvidos para este sistema são o comportamento de seguir o objetivo e o de seguir linha, onde o primeiro busca seguir a trajetória planejada, sendo sua resposta resultantes as velocidades calculadas, enquanto o segundo comportamento busca seguir a linha, se adaptando às mudanças da pista ou do ambiente e à falhas do robô. O segundo comportamento utiliza a percepção do ambiente

para determinar a posição do robô em relação à linha, este erro de posição é ajustado através de um ganho K_R, e assim, são definidas as velocidade necessárias para corrigir a posição do robô.

Os pesos utilizados no Controlador de Schemas foram definidos através de simulações, onde se procurou obter o menor erro na execução da trajetória sem sair da pista, e depois de sua estipulação, os pesos foram testados num experimento utilizando o ASURO. Os resultados obtidos, tanto nas simulações quanto nos testes, mostraram-se satisfatórios, apesar disso, observou-se que o resultado da parte reativa (velocidades do comportamento de Seguir linha) pode ser melhorado através de outros ajustes do erro de posição, sendo utilizados outros métodos de controle (PI, PD, PID, robusto, adaptativo, ...) ou métodos de estimação.

Portanto, observou-se que o sistema de navegação proposto mostrou-se robusto e eficiente, sendo apto a ser utilizado em diversas aplicações, além disso, o uso da arquitetura AuRA torna o sistema adaptável, permitindo mudanças ou melhorias nas técnicas utilizadas. Assim, a utilização e aperfeiçoamentos deste sistema mostram-se interessantes para o desenvolvimento de uma navegação robusta e confiável para as aplicações futuras imaginadas.

Futuramente, além das melhorias descritas anteriormente, tais como a implementação de outro método no ajuste de erro no cálculo das velocidades reativas, é interessante a implementação de técnicas de SLAM, de desvio de obstáculos e aperfeiçoar a abordagem utilizada no Planejamento da Missão, tornando-a mais dinâmica e automática.

Referências

ALBUS, James. Brains, Behavior, and Robotics, BYTE Books, Peterborough, NH. 1981. 400p.

ALBUS, J.; PROCTOR, F. A reference model architecture for intelligent hybrid control systems. In: Proceedings of the 1996 Triennial World Congress, International Federation of Automatic Control (IFAC). San Francisco, CA: [s.n.], 1996. Disponível em: <http://www.isd.cme.nist.gov/documents/albus/ifac13.pdf>. Acesso em: abr.30, 2010.

ALVES NETO, Armando. Geração de Trajetórias para veículos aéreos autônomos não-tripulados. 2008. Dissertação (Mestrado) – Ciência da Computação, Universidade Federal de Minas Gerais, BeloHorizonte.

ANDERSON, David P.; HAMILTON, Mike. The Journey Robot. Disponível em: http://www.geology.smu.edu/~dpa-www/robo/jbot/. Acessado em: abr. 23, 2010.

ARBIB, Michael. A. Schema theory. In: SAPHIRO, S. (Ed.). The Encyclopedia of Artificial Intelligence. 2. ed. New York, NY: Wiley-Interscience, 1992. p. 1427–1443

ARKIN, Ronald C. Behavior-Based Robotics. Cambridge: MIT Press, 1998, 491p.

ARKIN, R. C. Towards the unification of navigational planning and reactive control. In: Working notes of the AAAI 1989 Spring Symposium on Robot Navigation. Stanford University: [s.n.], 1989. Disponível em: <http://www.cc.gatech.edu/ai/robot-lab/onlinepublications/ stanford.pdf>. Acesso em: abr.22, 2010.

ASUROWIKI. Disponível em: http://www.asurowiki.de/pmwiki/pmwiki.php. Acessado em: out.11, 2009.

BERGLUND, T.; JONSSON, H; SDERKVIST, I. An obstacle-avoiding minimum variation b-spline problem. Proceedings of the 2003 International Conference on Geometric Modeling and Graphics, 2003.

BIANCHI, Reinaldo. Introdução à Robótica: Aulas dadas ao Mestrado em Inteligência Artificial Aplicadas a Automação. São Bernardo do Campo, 2007. Disponível em: <http://www.fei.edu.br/~rbianchi/robotica/ >. Acesso em: 20 nov. 2008.

BIGDOG. Disponível em: http://www.bostondynamics.com/robot_bigdog.html. Acessado em: abr.23, 2010.

BROOKS, Rodney. A robust layered control system for a mobile robot. Robotics and Automation, IEEE Journal of, v.2, n.1, pp. 4–23, mar 1986.

CHARNIAK, Eugene; MCDERMOTT, Drew V. Introduction to artificial intelligence. Reading MA: Addison-Wesley, 1985, 701 p.

CHOI, J.; CURRY, R. E.; ELKAIM, G. H. Continuous Curvature Path Generation Based on Bézier Curves for Autonomous Vehicles. IAENG International Journal of Applied Mathematics, 40:2, IJAM_40_2_07. (Advance online publication: 13 May 2010).

CIM. Disponível em: http://www.cim.mcgill.ca/~junaed/robotics.html. Acessado em: abr.23, 2010.

CLARKE, R. Asimov's Laws of Robotics: Implications for Information Technology. Chapman ACT, 1994. Disponível em: <http://www.rogerclarke.com/SOS/Asimov.html>. Acessado em: nov.8, 2009.

CONNELL, Jonathan. H. A Colony Architecture for an Artificial Creature. MIT AI Laboratory, Cambridge, MA, August 1989.

CORMEN, Thomas. Introduction to algorithms (2ª ed.). MIT Press, Cambridge, MA,2001. 1216p.

CROWLEY, James L. Navigation for an Intelligent Mobile Robot. Tech. Report CMU-RI-TR-84-18, Robotics Institute, Carnegie Mellon University, 1984. 25p.

DEFENSE UPDATE. Mini-Robots Operating Indoor. Disponível em: http://defense-

update.com/features/2008/november/11208_minirobotugv_urbancombatindoor.html . Acessado em: abr. 23, 2010.

DEVICEGURU. Build your own Rubik's Cube-solving robot. 2008. Disponível em: http://deviceguru.com/build-your-own-rubiks-cube-solving-robot/. Acessado em: out. 10, 2008.

DRROBOT. Sputnik. 2008. Disponível em: http://www.drrobot.com/products_item.asp?itemNumber=SPUTNIKAcessado em: out. 10, 2008.

DU PLESSIS, L. J. ;SNYMAN, J. H. Trajectory-planning through interpolation by overlapping cubic arcs and cubic splines. International Journal for Numerical Methods in Engineering. Int. J. Numer. Meth. Engng; v. 57, pp.1615–1641, 2003

DUBINS, L.E. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents, American Journal of Mathematics, v.79, pp. 497–516, 1957.

DUDEK, G., FREEDMAN, P. ;HADJRES, S. Using local information in a non-local way for mapping graph-like worlds. in Proc. of the 13th Int. Joint Conf. on Artificial Intelligence (IJCAI-93), pp. 1639–1645, 1993.

DUDEK, Gregory; JENKIN, Michael, Computational principles of mobile robotics, Cambridge University Press, New York, NY, 2000.

FESTO. The complete Robotino® package. 2008. Disponível em: http://www.festo-didactic.com/int-en/learning-systems/education-and-research-robots-robotino/the-complete-robotino-package.htm?fbid=aW50LmVuLjU1Ny4xNy4xOC44NTguNDc1NgAcessado em: out. 10, 2008.

FIKES, R. E.; NILSSON, N. STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, v. 5, n. 2, p. 189–208, 1971.

GIRALT, G.; CHATILA, R.; VAISSET, M. An integrated navigation and motion control system for autonomous multisensory mobile robots. In: BRADY, M.;

PAUL, R. (Ed.). First International Symposium on Robotics Research. Cambridge,MA:MIT Press, 1984. p. 191–214.

GIZMO WATCH. WowWee's Rovio: The 3-wheeled robotic home surveillance. 2008. Disponível em: http://www.gizmowatch.com/entry/wowwees-rovio-the-3-wheeled-robotic-home-surveillance/. Acessado em: out. 10, 2008.

GIZMODO. Disponível em: < http://gizmodo.com/5366082/250+foot-long-hybrid-airship-will-spy-over-afghanistan-battlefields-in-2011>. Acessado em: ago. 22, 2010.

GRASSI JR, Valdir. Arquitetura Híbrida para Robôs Móveis Baseada em Funções de Navegação com Iteração Humana, 2006, 118p. Tese (Doutorado) – Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos, Universidade de São Paulo, São Paulo.

HEINEN, Farley José. Sistema de Controle híbrido para robôs móveis autônomos. 2002, 130p. Dissertação (Mestrado) – Centro de Ciências Exatas e Tecnológicas, Universidade do Vale do Rio dos Sinos, São Leopoldo.

HUSQVARNA.Disponível em: <www.husqvarna.com/br>. Acessado em: abr. 22, 2010.

IBRAHIM, Y.; FERNANDES, A. Study on Mobile Robot Navigation Techniques. IEEE International Conference on Industrial Technology (ICIT), v.1, pp. 230-236, 2004.

IROBOT. Disponível em: <http://www.irobot.com/>. Acessado em: abr. 22, 2010.

IRT. Information and Robot Technology at the University of Tokyo Press Release. Disponível em:<http://www.irt.i.u-tokyo.ac.jp/news/pressrelease081024_e.pdf>. Acessado em: mar.10, 2010.

ITMB. Mobile Robots. 2008. Disponível em: http://www.itmb.gr/equipment/MobileLab.html. Acessado em: out. 10, 2008.

IYENGAR, S. S.; ELFES, A. Autonomous Mobile Robots: Control, Planning, and Architecture. Los Alamitos, California: IEEE Computer Society Press, 1991.

JEFFERIS, David. Robot Workers. New York: Crabtree Pub.2006, 32 p.

KAELBLING, Leslie. An Architecture for Intelligent Reactive Systems. SRI International Technical Note No. 400, Menlo Park, CA, October. 1986.

KALIPEDIA ROBOTICA. Disponível em: http://ar.kalipedia.com/tecnologia/tema/graficos-encoder-optico.html?x1=20070821klpinginf_48.Ees&x=20070821klpinginf_92.Kes&x2=20070821klpinginf_89.Kes. Acessado em: jun.22, 2010.

KOMORIYA, K.; TANIE, K. (1989). Trajectory Design and Control of a Wheel-type MobileRobot Using B-spline Curve. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 398–405, Tsukuba, Japan.

KORTENKAMP, Davi.; BONASSO, R. Peter.; MURPHY, Robin. Artificial intelligence and móbile robots: case studies of successful robot systems. Menlo Park, California: The AAAI Press/The MIT Press, 1998.

KOYUNCU, E.; INALHAN, G. A probabilistic b-spline motion planning algorithm for unmanned helicopters flying in dense 3d environments. In International Conference Intelligent Robots and Systems, Nice, 2008.

KUIPERS, B. A Robust, Qualitative Method for Robot Spatial Learning. Proc. of the Seventh National Conference on Artificial Intelligence, pp. 774-779, Saint Paul, Minnesota. 1988.

LATOMBE, Jean Claude. Robot Motion Planning. Norwell, MA. Kluwer Academics Publishers, 1991. 654p.

LAVALLE, Steven. Planning Algorithms. New York, NY Cambridge University Press ,2006. 852p.

LESTER, Patrick. A* Pathfinding for Beginners. 2005. Disponível em: http://www.policyalmanac.org/games/aStarTutorial.htm. Acessado em: mai. 15, 2009.

LEVI, P. Principles of planning and control concepts for autonomous mobile robots. In proceedings.of IEEE International Conference on Robotics and Automation. , pág. 874, 1987.

LIMA, Carlos Raimundo Erig. Proposta de ambiente baseado em computação reconfigurável para aplicação em protótipos de sistemas embarcados, 2003, 214p. Tese (Doutorado) – Faculdade de Engenharia Mecânica, Universidade estadual de Campinas, Campinas.

MACKENZIE, D.; CAMERON, J.; ARKIN, R. Specification and execution of multiagent missions. In: Proceedings of the International Conference on Intelligent Robotics and Systems (IROS). Pittsburgh, PA: IEEE Computer Society, 1995. p. 51–58.

MAES, P. The dynamics of action selection. In: Proceedings of the Eleventh International Joint Conference on Articial Intelligence (IJCAI89). Detroit, MI: [s.n.], 1989. v. 2, p. 991–998.

MELO, Leonimer Flávio de. Proposta de Simulador Virtual para Sistemas de Navegação de Robôs Móveis Utilizando Conceitos de Prototipagem Rápida, 2007. 293p. Tese (Doutorado) – Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, Campinas.

MEYSTEL, A. Knowledge based nested hierarchical control. In: SARIDIS, G. (Ed.). Advances in Automation and Robotics. [S.l.]: JAI Press, 1990. v. 2, p. 63–152.

MISHKIN, Andrew. Sojourner: an insider's view of the Mars Pathfinder mission. New York: Berkley Books, 2003, 333 p.

MOON, I.; MIURA, J.; SHIRAI, Y. Automatic Extraction of Visual Landmarks for a Mobile Robot Under Uncertainty of Vision and Motion. Proc. of IEEE Int. Conf. on Robotics and Automation, pp. 1188-1193, Korea. 2001.

MORAVEC, H. P. Towards automatic visual obstacle avoidance. In: Proceedings of the Fifth International Joint Conference on Artificial Intelligence. Cambridge, MA: [s.n.], 1977. p. 584.

MORAVEC, H. P. The stanford cart and the CMU rover. In: Proceedings of the IEEE. [S.l.: s.n.], 1983. v. 71, p. 872–884.

MURRAY, R. M., LI,Z. e SASTRY S. S. A Mathematical Introduction to Robotic Manipulation. CRC Press, 1994. 474p. Disponível em: http://www.cds.caltech.edu/~murray/books/MLS/pdf/mls94-complete.pdf. Acessado em: ago. 11, 2010.

MURPHY, Robin. Introduction to AI Robotics. Cambridge, Massachusetts: MIT Press, 2000. 496p.

NARA. GPS. Disponível em: http://www.nara.org.br/servicos/ntp/gps/. Acessado em: abr. 23, 2010.

NEHMZOW, Ulrich. Mobile Robotics: A Practical Introduction. Minneapolis: Springer, 2000. 280p.

NOURBAKHSH, I. R.; KUNZ, C.; WILLEKE, T. The Mobot museum robot installations: a five year experiment. In Proc. IROS 2003 (2003), 3636--3641.

OTTONI, G. L.; LAGES, W. F.. Navegação de Robôs Móveis em Ambientes Desconhecidos utilizando Sonares de Ultra-som. Revista Controle & Automação/Vol.14 no.4/Outubro, Novembro e Dezembro, PP. 402-411, 2003.

PARK, P.; KENDER, J. R. Topological Direction-Giving and Visual Navigation in Large Environments. Artificial Intelligence 78(1-2): 355-395, 1995.

PEDROSA, Diogo Pinheiro Fernandes. Sistema de Navegação para Robôs Móveis Autônomos, 2001. Dissertação (Mestrado) - Engenharia Elétrica, Universidade Federal do Rio Grande do Norte, Natal.

PICASA. ASURO. 2008. Disponível em: http://picasaweb.google.com/lh/photo/EbJ1Bc9SYHmFwmgh8-0hPgAcessado em: out. 10, 2008.

PIERI, Edson Roberto de. Curso de Robótica Móvel - UFSC. Florianópolis, 2002.

PODER AÉREO. MQ-9 ‘Reaper’ faz primeiro ataque no Iraque. Disponível em: http://www.aereo.jor.br/page/255/. Acessado em: abr. 23, 2010.

PROTECTOR. The Protector USV: delivering proven anti-terror and force protection capabilities. Disponível em: http://www.ws-wr.com/epk/BAE_Protector/. Acessado em: abr.23, 2010.

REMOLINA, E.; KUIPERS, B. Towards a general theory of topological maps. Artificial Intelligence, vol. 152, no. 1, pp. 47–104, 2004.

RIBEIRO, Monael Pinheiro. CAMIN – Cadeira Móvel Inteligente: Um Sistema de Navegação Utilizando uma Adaptação do Método dos Campos Potenciais Artificiais, 2007. Dissertação (Mestrado) – Ciência em Sistema e Computação, Instituto Militar de Engenharia, Rio de Janeiro.

RIBEIRO, C. H. C.; COSTA, A. H. R.; ROMERO, R. A. F. Robôs móveis inteligentes: Princípios e técnicas. In: MARTINS, A. T.; BORGES, D. L. (Ed.). Anais do XXI Congresso da Sociedade Brasileira de Computação. Fortaleza: SBC, 2001. v. 3, p. 257–306.

ROBÓTICA-12C. Dísponível em: http://robotica-12c.comyr.com/Robotica%20no%20Qotidiano.htm. Acessado em: abr. 23, 2010.

ROSHEIM, Mark Elling. Leonardo´s Lost Robots. Minneapolis: Springer, 2006, 184 p.

ROSENBLATT, J. DAMN: A distributed architecture for mobile navigation. Journal of Experimental and Theoretical Artificial Intelligence, v. 9, n. 2 / 3, p. 339 – 360, 1997. Disponível em: <http://www.ri.cmu.edu/pubs/pub_3134.html>. Acesso em: fev.19, 2010.

ROSENSCHEIN, Stanley; KAELBLING, Leslie. The Synthesis of Digital Machines with Provable Epistemic Properties. SR1 International Technical Note No. 412, Menlo Park, CA, 1987.

RUSSELL, Stuart J.; NORVIG, Peter. Artificial Intelligence: A Modern Approach. Prentice-Hall, 2003. 912p.

SANTOS, Kleber Roberto da Silva. Sistema de Navegação Autônoma para Robôs Móveis baseado em Arquitetura Híbrida: Teoria e Aplicação, 2009. Dissertação (Mestrado) – Engenharia Elétrica, Universidade Federal de Itajubá, Itajubá.

SICILIANO, Bruno; KHATIB, Oussama. Springer Handbook of robotics. Berlin: Springer, 2008. 1611 p.

SIEGWART, Roland; NOURBAKHSH, Illah R. Introduction to Autonomous Mobile Robots. Cambridge: MIT Press, 2004. 321 p.

SHANMUGAVEL, M. Path Planning of Multiple Autonomous Vehicles, 2007. PhD – College of Management and Technology, Cranfield University, Shrivenham.

SHARMA, S.; KULCZYCKI, E. A.; ELFES, A. Trajectory Generation and Path Planning for Autonomous Aerobots in Proceedings book of International Conference on Robotics and Automation, (Roma , Italy), 2007.

TERRA TECNOLOGIA. Polícia francesa testa robô voador para segurança. 2008. Disponível em: http://tecnologia.terra.com.br/interna/0,,OI2643588-EI8328,00.html. Acessado em: abr.22, 2010.

THE TRIBUNE. Lifesaving Robots. Disponível em: http://www.tribuneindia.com/2003/20030602/login/main2.htm. Acessado em: 1br.23, 2010.

THRUN, Sebastian; FOX, Dieter; BURGARD, Wolfram. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31:29-53, 1998. Também aparece em Autonomous Robots 5, 253-271 (joint issue).

TOP NEWS. Canadian technologists develop world's first flying micro-robot. 2009. Disponível em: http://www.topnews.in/canadian-technologists-develop-worlds-first-flying-microrobot-2150208&usg=__-aHXXsFpggTRxNZNqDDV5_eGf6s=&h=294&w=400&sz=45&hl=pt-BR&start=1&um=1&itbs=1&tbnid=VSPFGo44NOgr6M:&tbnh=91&tbnw=124&prev=/images%3Fq%3Dmicrorobot%26um%3D1%26hl%3Dpt-BR%26sa%3DX%26tbs%3Disch:1. Acessado em: abr. 17, 2010.

VUKOBRATOVIC, M. K. When were active exoskeletons actually born? Humanoid Robotics, International Journal of, v.4, n.3, pp.459–486, jan 2007.

WALTHER, M.; STEINHAUS, P.; DILLMANN, R. Using B-splines for mobile robot path representation and motion control. Disponível em: < http://wwwiaim.ira.uka.de/data/File/Publications/using%20b-splines.pdfl>. Acessado em: mar.8, 2010.

WEISSTEIN, Eric W. B-Spline. From MathWorld--A Wolfram Disponível em: http://mathworld.wolfram.com/B-Spline.html. Acessado em: abr. 25, 2010.

WIKIMEDIA COMMONS. Sojourner on Mars PIA01122.jpg. 2005. Disponível em: http://commons.wikimedia.org/wiki/File:Sojourner_on_Mars_PIA01122.jpg. Acessado em: ago. 21, 2009.

WIKIMEDIA COMMONS. Shakey.png. 2007. Disponível em: http://commons.wikimedia.org/wiki/File:Shakey.png. Acessado em: ago. 21, 2009.

ANEXO A – Arquiteturas de Controle

O estudo de arquiteturas de controle para robôs móveis existe desde a construção do primeiro robô móvel (ARKIN, 1998), pois a arquitetura consiste na estruturação dos componentes computacionais existentes num robô móvel de modo que esses componentes possam ser modificados sem que o sistema robótico falhe.

Existem diversas definições de arquitetura de robôs móveis. Segundo Bonasso, uma arquitetura de robôs móveis refere-se à organização do software de controle do robô. (KORTENKAMP, BONASSO e MURPHY, 1998). Albus (1981) acredita que a arquitetura é a descrição de como um sistema é construído a partir de componentes básicos e como estes componentes se encaixam formando o todo. E, Arkin (1998), descreve a arquitetura robótica como uma disciplina dedicada ao design de robôs altamente específicos e individuais a partir coleção de blocos de construção comum de software.

De forma geral, os componentes básicos encontrados em um sistema de controle, que são utilizados para definir uma arquitetura para robôs, podem ser classificados em três grupos principais (IYENGAR; ELFES, 1991):

(1) Percepção: envolve as atividades de interpretação dos sensores, integração dos sensores, modelagem do mundo real, e reconhecimento;

(2) Planejamento: que envolve o planejamento de tarefas, a sincronização, e o monitoramento da execução de toda a atividade do robô; e

(3) Atuação: que envolve as atividades de execução dos movimentos e ações do robô, e controle dos atuadores.

Quanto à classificação de uma arquitetura para robôs, segundo à abordagem utilizada no desenvolvimento da arquitetura, que pode ser funcional, comportamental, ou uma abordagem mista. Nas arquiteturas com abordagem funcional, podem-se identificar módulos ou componentes que possuem funções definidas dentro da arquitetura, tais como localização, mapeamento, planejamento, etc. Nas arquiteturas que utilizam uma abordagem

comportamental, a arquitetura é formada principalmente por componentes chamados comportamentos, responsáveis pela realização de uma tarefa determinada. Assim, os módulos existentes numa abordagem funcional estão divididos por funções, enquanto os módulos de uma abordagem comportamental estão divididos por ações e tarefas que o robô deve realizar. E, arquiteturas com abordagem mista são arquiteturas desenvolvidas procurando combinar as duas abordagens (Funcional/Comportamental) (GRASSI JR., 2006).

A função pode ser definida como utilidade de algo, no caso da robótica móvel, é a utilidade de um módulo para o robô. Enquanto, comportamento pode ser definido como uma função que relaciona estímulos sensoriais a ações produzidas sobre os atuadores do robô, de acordo com um plano realizado a partir de um modelo interno do ambiente (RIBEIRO, COSTA e ROMERO, 2001).

De acordo com esta compreensão, dependendo da complexidade do plano e modelo interno utilizado, os comportamentos podem ser classificados em uma escala gradual que vai de comportamentos puramente reativos, que não utilizam um plano e modelo interno do mundo, até comportamentos deliberativos complexos. Esta compreensão de comportamento geralmente é utilizada no contexto de arquiteturas híbridas. Às vezes, neste contexto, comportamentos podem também ser chamados de controladores ou habilidades (GRASSI JR., 2006).

Outra forma de se classificar as arquiteturas de controle está relacionada ao uso de deliberação e reatividade dentro do sistema de controle. Segundo este critério, as arquiteturas podem ser divididas em: deliberativas, reativas, e híbridas (MURPHY, 2000). Nas seções a seguir serão apresentadas as arquiteturas de controle seguindo esta classificação, e por último será detalhada a arquitetura AuRA, arquitetura utilizada no trabalho.

A.1 Arquitetura Deliberativa

A arquitetura deliberativa é uma abordagem que procura imitar o processo de planejamento e tomada de decisão do homem (GRASSI JR., 2006), pois, na arquitetura deliberativa, os robôs utilizam o mesmo princípio de decisão, ilustrado na Figura A.1, que baseia-se em perceber o ambiente, planejar uma ação e agir, e dessa forma, a partir de um modelo interno de representação do mundo, o robô adota as ações necessárias para a realização de um objetivo. No sistema de controle do robô, as informações sobre o ambiente em que o robô esta inserido são armazenadas num modelo interno do mundo, formado a partir do conhecimento a priori sobre o ambiente e de informações adquiridas pelos sensores do robô.

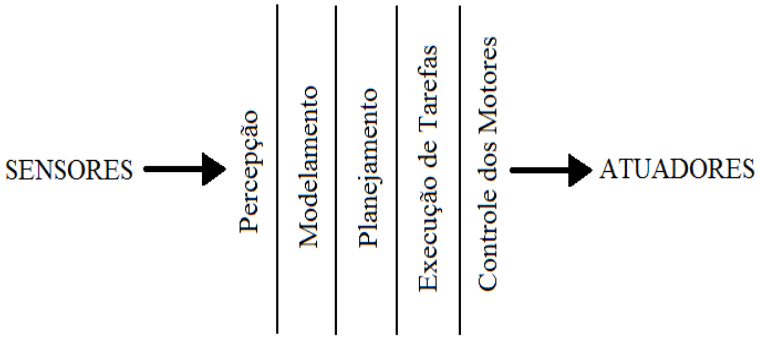


Figura A.1 – Arquitetura Deliberativa.

O modelo interno do mundo pode ser representado de diversas formas, entre elas, as mais utilizadas são o modelo simbólico e o modelo geométrico. O modelo do tipo simbólico é baseado em lógica, sendo tradicionalmente utilizado em inteligência artificial. O modelo geométrico representa o ambiente de forma espacial indicando regiões livres e regiões ocupadas por obstáculos. Entretanto, independente da representação do modelo do mundo, as ações a serem realizadas pelo robô para alcançar de objetivo são definidas a partir deste modelo interno do mundo. Por exemplo, no caso de modelos simbólicos, o objetivo pode ser expresso por um estado que o mundo, incluindo o robô, deve assumir, e no caso do modelo geométrico, o objetivo é representado por uma configuração que o robô deve assumir em uma tarefa de navegação.

Os primeiros trabalhos em robótica móvel utilizavam uma abordagem puramente deliberativa, um exemplo é o Shakey que utilizava um modelo interno simbólico do mundo e um planejador chamado STRIPS (FIKES; NILSSON, 1971). Os robôs HILARE (GIRALT; CHATILA; VAISSET, 1984), o Stanford Cart e o CMU Rover (MORAVEC, 1977, 1983), também possuíam sistemas de controle deliberativos, mas que se baseavam em modelos internos geométricos do ambiente (GRASSI JR., 2006).

O planejamento de um robô móvel consiste em utilizar o modelo interno para determinar suas ações a fim de atingir um objetivo, assim, como estas ações dependem do modelo do mundo, faz-se necessário que o modelo interno seja o mais consistente, confiável, preciso, atual e completo possível. Se as informações contidas no modelo estiverem imprecisas ou defasadas, as ações planejadas podem descumprir não alcançar o objetivo estipulado.

Assim, as mudanças no ambiente devem ser detectadas pelo robô durante a execução, atualizando o modelo interno do robô, e então, re-planejando as suas ações. Portanto, devido à necessidade de constante atualização e re-planejamento, as arquiteturas deliberativas são mais adequadas para ambientes praticamente estáticos e muito bem controlados. Em ambientes dinâmicos, onde mudanças podem ocorrer a qualquer momento, o modelo interno irá mudar constantemente, necessitando de re-planejamento constante, tornando impraticável o uso deste tipo de arquitetura.

A arquitetura deliberativa pode ser descrita como mostrado no esquema da Figura A.1 (ARKIN, 1998), onde pode observar que o funcionamento básico deste tipo de arquitetura dá-se de forma que após a percepção do mundo, o robô tenta atualizar seu modelo do mundo e então toma a decisão do que fazer. Dentre os diversos tipos de arquitetura deliberativa é possível destacar duas: a arquitetura (NHC) desenvolvida por Meystel (1990) e a arquitetura NIST Realtime Control System (RCS) desenvolvida por Albus e Proctor (1996).

A.1.1 Nested Hierarchical Controller (NHC)

Na arquitetura NHC (MEYSTEL, 1990), esquematizada na Figura A.2, o robô coleta informações sensoriais e combina estas informações em uma estrutura de dados que representa o modelo do mundo. As informações adquiridas podem ser combinadas no modelo do mundo juntamente com conhecimentos fornecidos a priori. Esta atividade de construção e atualização deste modelo faz parte da atividade de sensoriamento, ou percepção, dentro da arquitetura (MURPHY, 2000).

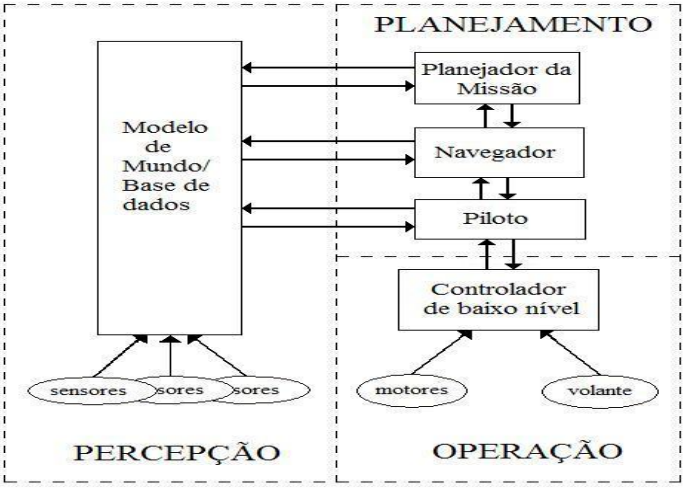


Figura A.2 – Arquitetura NHC.

A maior contribuição da arquitetura NHC é a divisão do planejamento em 3 sub-sistemas: o Planejador da Missão, o Navegador e o Piloto. Estes três módulos de planejamento são executados seqüencialmente, do nível de abstração mais alto, o Planejador da missão, até o nível de abstração mais baixo, o Piloto. Cada módulo executa uma determinada tarefa, o Planejador da Missão traduz os comandos enviados ao robô e determina a posição atual do robô e dos objetivos a serem alcançados; o Navegador, utilizando as informações determinadas pelo Planejador da Missão, encontra o trajeto necessário para atingir o objetivo e gera um conjunto de waypoints (sub-objetivos); o Piloto recebe o primeiro waypoint e determina as ações que o robô deverá realizar, e conforme for alcançando cada waypoints, o Piloto recebe o próximo waypoint até alcançar o objetivo final (MURPHY, 2000).

Conforme o robô se move, o modelo interno do mundo é atualizado, porém, o ciclo completo de planejamento não se repete. Quando necessário, o Piloto corrige o movimento local do robô para lidar com eventuais divergências percebidas. Se não for suficiente, então o Navegador gera outra trajetória, e somente em último caso, o Planejador da missão refaz o planejamento da missão (GRASSI JR., 2006).

A.1.2 NIST Realtime Control System (RCS)

A arquitetura RCS (ALBUS e PROCTOR, 1996) foi criada para servir como um guia para fabricantes que queiram adicionar mais inteligência a seus robôs. Ela foi baseada na arquitetura NHC, no entanto foram introduzidas algumas modificações (GRASSI JR., 2006).

A arquitetura RCS, esquematizada na Figura A.3, possui a mesma hierarquia no planejamento que a arquitetura NHC, contudo, como observado nas Figuras A.2 e A.3, há diferenças significativas entre elas, que consistem num pré-processamento das informações sensoriais, extraindo informações em diferentes níveis de abstração para serem integradas no modelo do mundo; um módulo de julgamento de valor, que simula os planos de ação para verificar se estes satisfazem os requisitos da tarefa; e o módulo de planejamento envia o plano para outro módulo, o Gerador de Comportamentos, que converte o plano em ações, este módulo é similar ao módulo Piloto da arquitetura NHC, mas parece haver uma menor orientação para tarefas de navegação. Há outro módulo, que não é possível visualizar no esquema, a Interface com o Operador, que permite o humano observar e interagir com o programa (MURPHY, 2000).

A RCS permite a interação de um operador humano nas diversas atividades organizadas nos vários níveis hierárquicos da arquitetura, prevendo a possibilidade do homem compartilhar o controle com o sistema. Assim, em qualquer nível de hierarquia, o homem pode substituir completamente as atividades do sistema com sua capacidade de percepção, tomada de decisões e controle na execução de tarefas. Isso permite que a

autonomia no sistema possa ser desenvolvida de forma incremental, enquanto as atividades dos níveis superiores da arquitetura são realizadas pelo homem. Conforme a disponibilidade de tecnologia em robótica, o robô poderá executar tarefas de maior responsabilidade e de níveis hierárquicos maiores até atingir um grau completo de autonomia (GRASSI JR., 2006).

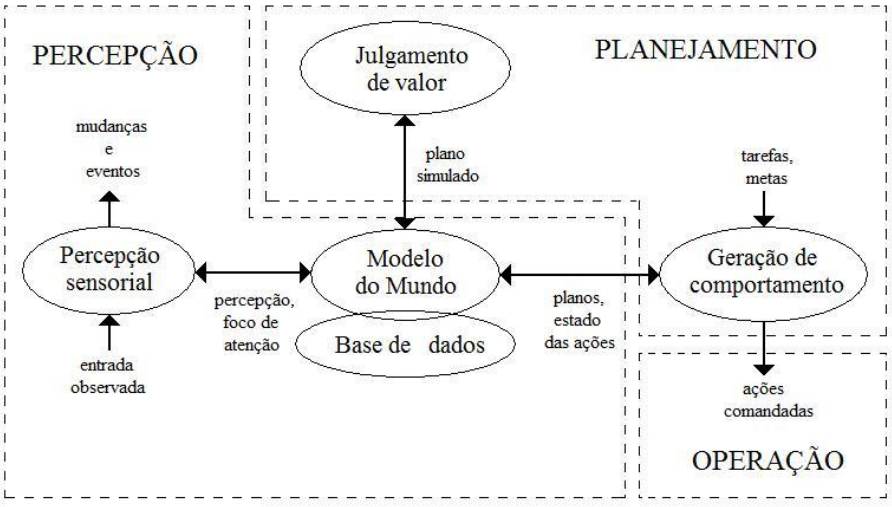


Figura A.3 – Arquitetura RCS..

A.2 Arquitetura Reativa

Em 1986, Brooks atraiu a atenção para a utilização de arquiteturas reativas na robótica móvel, apresentando a arquitetura Subsumption (BROOKS, 1986). Segundo ele a arquitetura deliberativa não era satisfatória para atender a dinamicidade do mundo real, e indicou a utilização de arquiteturas reativas, que evitam o uso de um modelo interno do mundo, afirmando que o mundo é o seu melhor modelo.

A principal motivação das arquiteturas reativas é permitir a implementação de sistemas de controle que possam responder de forma rápida a uma variedade de eventos ou situações no ambiente, fazendo com que robôs possam operar em ambientes extremamente dinâmicos (GRASSI JR., 2006). E, devido à simplicidade no tratamento das informações sensoriais e a rapidez em que uma ação é associada a uma percepção, a velocidade de processamento e resposta dos sistemas de controle reativos é alta, operando

satisfatoriamente em ambientes dinâmicos como um corredor, o trânsito ou campos esportivos. A Figura A.4 ilustra o conceito do funcionamento das arquiteturas reativas.

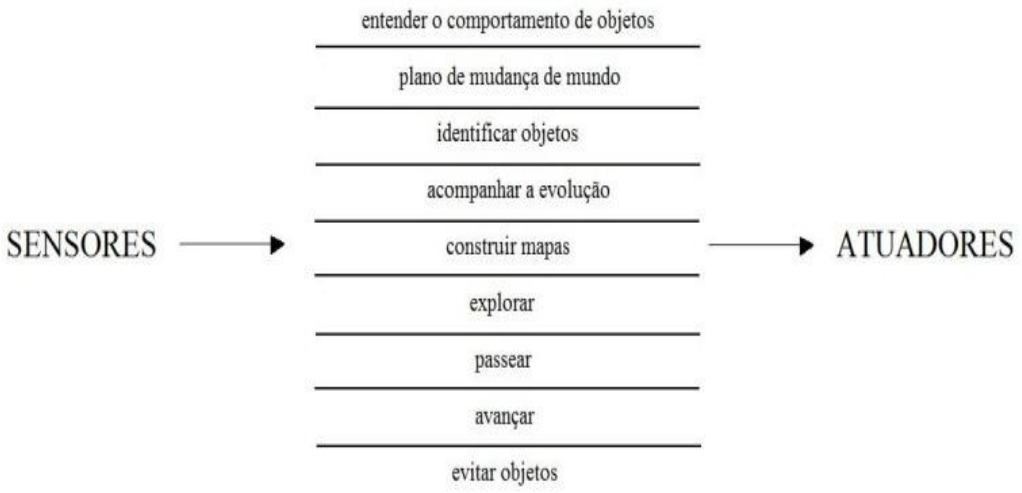


Figura A.4 – Conceito da arquitetura reativa.

A arquitetura reativa, por não utilizar um modelo interno do ambiente, geralmente assume que as características do ambiente que possam ser do interesse do robô para a realização da tarefa estejam sempre visíveis (ROSENBLATT, 1997). Dessa forma, o sistema de controle depende fortemente das informações locais obtidas por meio dos sensores do robô, ou seja, como o robô percebe o ambiente em que está inserido num determinado momento (GRASSI JR., 2006). Assim, devido à importância ao tempo de resposta do robô às mudanças no ambiente, a arquitetura reativa, esquematizada na Figura A.4, é organizada de forma que as ações do robô são decididas rapidamente a partir de respostas pré-definidas a determinadas informações sensoriais. Estes pares de percepção-ação, ou estímulo-resposta geralmente são chamados de comportamentos reativos (ilustrados na Figura A.5).

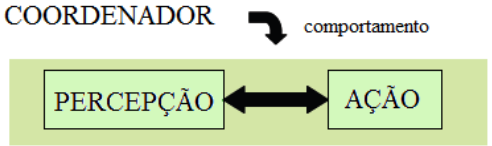


Figura A.5 – Comportamentos reativos.

A coordenação dos comportamentos em uma arquitetura reativa pode ser feita de forma competitiva ou cooperativa. Na coordenação competitiva, todos os pares percepção-ação ativos competem entre si, onde apenas um deles se sobressai, decidindo a ação que o robô deve realizar. Para determinar qual dos comportamentos possui prioridade, é necessário possuir um sistema para a escolha, pode ser por hierarquia, por seleção ou outra regra específica. Já na coordenação cooperativa, todos os comportamentos ativos contribuem para determinar a ação do robô. Um exemplo é o método utilizado nos motor-schemas (ARKIN, 1989), onde a ação resultante é determinada pela soma vetorial de todos os vetores de cada comportamento ativo.

A forte dependência às informações obtidas no momento da tomada de decisão pode tornar a utilização da arquitetura reativa muito restritiva, dependendo da aplicação e tarefa planejada, pois, às vezes se faz necessário o armazenamento temporário de informações sobre o ambiente. Até mesmo porque os sensores e os algoritmos que processam os dados sensoriais não estão livres de falhas e de ruídos que podem ser determinantes na execução de uma tarefa. Por isso, o sistema pode se beneficiar de filtros que utilizam a combinação de informações obtidas a partir de leituras consecutivas dos sensores dentro de um intervalo limitado de tempo (GRASSI JR., 2006).

Além das arquiteturas reativas já mencionadas, a subsumption (BROOKS, 1986) e a motor-schemas (ARKIN, 1989), existem outras arquiteturas reativas, das quais pode-se relevar a arquitetura de circuito (KAELBLING, 1986; ROSENSCHEIN e KAELBLING, 1987), a de seleção-ação (MAES, 1989), a arquitetura de colônia (CONNELL, 1989). A seguir será explicado resumidamente o funcionamento de cada uma destas arquiteturas.

A.2.1 Subsumption

A arquitetura subsumption, proposta por Brooks (1986), é uma das arquiteturas mais representativas dentro do paradigma puramente reativo (GRASSI JR., 2006). A arquitetura decompõe um complexo procedimento de um robô inteligente em módulos

simples de comportamentos, que são organizados em camadas, onde cada camada implementa um sub-objetivo particular, por exemplo, a decisão de seguir em frente.

Nesta arquitetura, os comportamentos são conectados uns aos outros formando uma rede organizada em camadas de competência. Cada camada é responsável por uma atividade do robô. Nas camadas mais altas estão os comportamentos responsáveis pelo cumprimento de uma tarefa específica que leva o robô a atingir um determinado objetivo, por exemplo, explorar o ambiente. Nas camadas mais baixas ficam os comportamentos responsáveis por ações básicas do robô, como, por exemplo, desvio de obstáculos (GRASSI JR., 2006).

Estes comportamentos são coordenados de forma competitiva, sendo que os comportamentos em camadas mais altas têm prioridade em relação aos comportamentos em camadas inferiores. Essa coordenação é feita por meio de dois mecanismos principais: inibição de entradas e supressão de saídas, como ilustra a Figura .A.6.

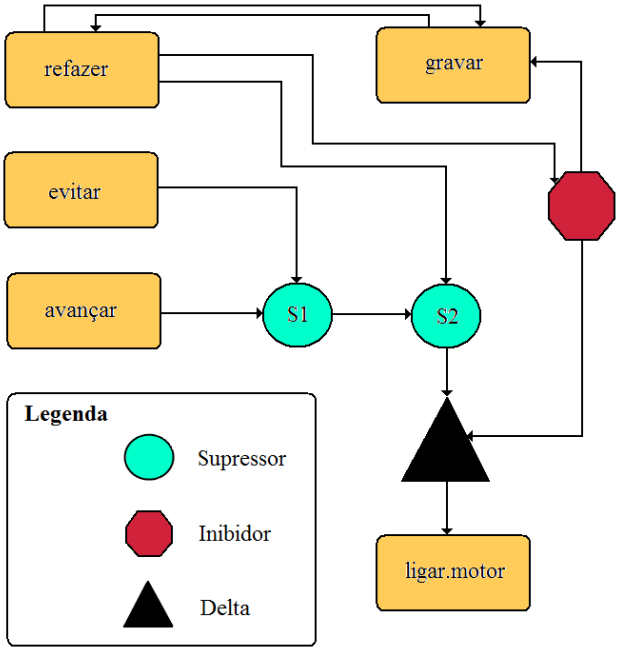


Figura A.6 – Arquitetura Subsumption.

Na supressão, a saída produzida pelo comportamento de prioridade mais baixa é suprimida pela saída produzida pelo comportamento de prioridade mais alta, e assim, a ação determinada pela camada superior é realizada. Entretanto, é necessário salientar, que ambos os comportamentos permanecem ativos. (PIERI, 2002).

Na inibição, o comportamento em nível mais baixo é desativado por aquele de nível mais alto, nesse caso, não ocorre uma substituição da ação de nível mais baixo, mas uma inibição do comportamento em si. (PIERI, 2002).

Na arquitetura de subsunção a hierarquia entre os comportamentos é definida de forma bastante específica. Um comportamento em uma camada superior só age inibindo ou suprimindo um determinado conjunto pré-definido de comportamentos em camadas inferiores, e não todos eles, assim, comportamentos que preservam a integridade física do robô e que se encontram nas camadas inferiores não são necessariamente suprimidos ou inibidos por comportamentos em camadas superiores (GRASSI JR., 2006).

A.2.2 Motor-Schema

Logo após o surgimento da arquitetura de subsunção, Arkin(1998) propôs um método para implementar comportamentos para robôs móveis baseando-se na biologia, utilizando a teoria de esquemas proposta por Arbib (1992).

Para Arkin as implicações que a teoria de esquemas de Arbib teriam para robótica autônoma são (ARKIN, 1998):

a) Os esquemas fornecem grande modularidade para expressar as relações entre o controle motor e da percepção, diferentemente dos modelos de rede neural.
b) Esquemas agem concorrentemente como cada um dos agentes distribuídos em uma cooperativa, ainda que de maneira competitiva e, portanto, são facilmente mapeados para processamento distribuído arquiteturas.

c) Os esquemas fornecem um conjunto de primitivas de comportamento pelo qual os mais complexos comportamentos podem ser construídos.
d) Suportes cognitivos e neurocientíficos existem para fundamentar esta abordagem. Estes podem ser modificados, se necessário, como adicionais modelos da neurociência ou cognitivos se tornem disponíveis.

No Motor-Schema, os comportamentos desejados são módulos que expressam a relação entre controle motor e percepção agindo de forma paralela e concorrente no sistema, cooperando uns com os outros para determinar a resposta geral do sistema (GRASSI JR., 2006).

A arquitetura Motor Schemas, esquematizada na Figura A.7, é construída de forma que cada esquema represente um comportamento desejado. A resposta de cada comportamento a uma percepção é representada na forma de um vetor com magnitude e orientação gerado a partir de um método de campos potenciais artificiais.

Estes vetores não são calculados para todo o ambiente, sendo calculados apenas para as informações relevantes para o comportamento na posição atual, por exemplo, no caso do comportamento de desvio de obstáculos, cada obstáculo criará um vetor com direção oposta ao obstáculo e magnitude em função da distância entre o robô e o obstáculo. Outro exemplo seria um comportamento para atingir o objetivo, onde este comportamento poderia um vetor em função da posição atual do robô e de uma posição de destino no ambiente, para atrair o robô para um objetivo de navegação.

Pode-se haver uma grande diversidade de comportamentos, tais como, mover-se em direção ao objetivo, desvio de obstáculos, manter-se na pista, seguir um líder, etc. Porém, a influência de um comportamento na resposta final é determinada por seu peso, que pode ser alterado para dar flexibilidade ao sistema.

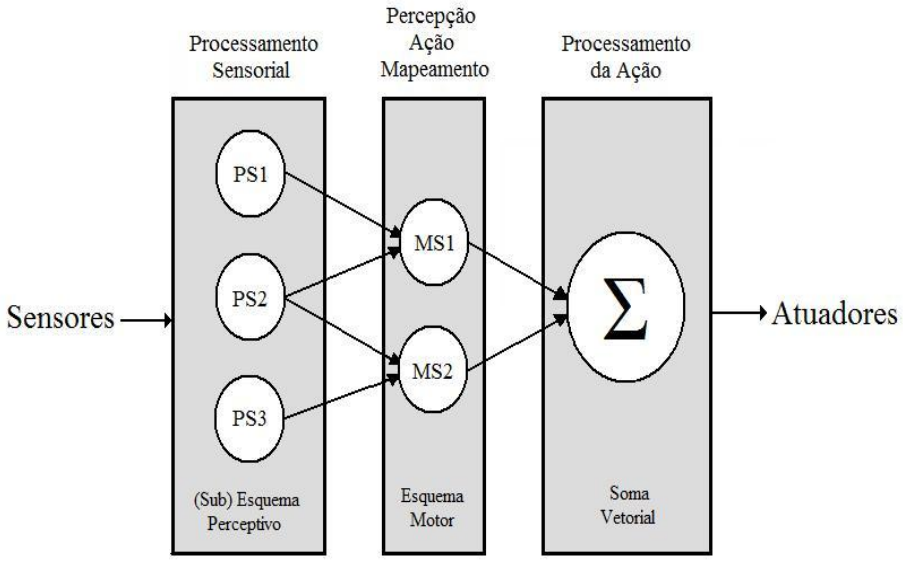


Figura A.7 – Arquitetura Motor-Schemas.

Além dos pesos, um comportamento também pode ter como parâmetro um nível de ativação, que determina quando o comportamento deve está ativo ou não. Com a possibilidade da combinação de diversos conjuntos de comportamentos primitivos é possível criar um comportamento emergente mais complexo.

O vetor resultante vai ser a soma de cada um dos vetores encontrados para os comportamentos ativos multiplicados pelos seus respectivos pesos. Obtendo-se assim, a direção e velocidade necessária para o robô atingir seu objetivo.

A arquitetura Motor Schema difere de outras abordagens comportamentais em váras maneiras significativas (ARKIN, 1998).

a) Respostas comportamentais são representadas em um formato uniforme: vetores gerados através de uma abordagem campos potenciais.
b) A coordenação é alcançada através de meios de cooperação com a adição de vetores.
c) Não existe hierarquia pré-definida pela coordenação, em vez disso, os comportamentos são configuradas em tempo de execução com base em

intenções do robô, capacidades e restrições ambientais. Os esquemas podem ser instanciados ou não instanciados em qualquer tempo com base em eventos perceptivos, portanto, a arquitetura é mais uma rede dinâmica do que uma arquitetura em camadas.

d) Arbitragem pura não é usada, em vez disso, cada comportamento pode contribuir em graus variados para a resposta geral do robô. A relação de forças entre os comportamentos (G) determinar a resposta global do robô.
e) Percepção de incerteza pode ser refletida na resposta do comportamento, permitindo-a servir como uma entrada no cálculo comportamental.

A.2.3 Arquitetura de Circuito

Arquitetura de Circuito, desenvolvida por Rosenschein e Kaelbling (1986), é uma hibridização dos princípios da reatividade como tipificado pela arquitetura de subsunção, utilizando abstrações e formalismos lógicos (ARKIN, 1998).

Uma vantagem que esta abordagem oferece envolve o uso de abstração através do agrupamento de comportamentos reativos e permitindo a arbitração dentro de cada nível de abstração, ou seja, o que os designers chamam de mediação hierárquica. Outra vantagem é o uso da lógica formal como um meio para expressar os comportamentos, permitindo a compilação em hardware e ajudando com a verificação do desempenho do sistema resultante de robótica (KAELBLING, 1986; ROSENSCHEIN e KAELBLING, 1987).

As motivações para esta arquitetura é típica para sistemas baseados em comportamento em geral: modularidade, permitindo o desenvolvimento incremental; sensibilização, forte acoplamento entre percepção e ação; e robustez, sendo capaz de operar apesar de circunstâncias imprevistas ou falhas no sensor. (ARKIN, 1998)

A.2.4 Action-Selection

A arquitetura Action-Selection desenvolvida por Maes (1989) utiliza um mecanismo dinâmico para selecionar os comportamentos que devem ser ativados. Ao invés

de utilizar uma estratégia pré-definida como na arquitetura de subsunção, os comportamentos individuais utilizam um nível de ativação para selecionar em tempo de execução quais comportamentos devem ativados. (ARKIN, 1998).

O nível de ativação é afetado pela situação atual em que o robô se encontra, pelos objetivos de alto nível, ou pela inibição causada por comportamentos conflitantes. Níveis de ativação também decaem com o passar do tempo. O comportamento com nível de ativação maior é escolhido dentre um conjunto de todos os comportamentos cujas pré-condições também estejam satisfeitas (ARKIN, 1998).

Devido à inexistência de uma organização clara dos comportamentos em camadas pré-definidas como na arquitetura subsumption, é difícil predizer qual o comportamento global emergente que o robô apresentará em um ambiente dinâmico, tendo uma ótima qualidade emergente (ARKIN, 1998).

A abordagem desta arquitetura também compartilha muito da filosofia com a teoria de esquema (ARBIB 1992), especialmente quanto ao uso dos níveis de ativação para controlar o desempenho comportamental. Sua principal limitação percebida é a falta de implementações reais em robôs reais e, portanto, não existe nenhuma evidência de quão facilmente os atuais formatos dos comportamentos individuais iriam realizar em tarefas robóticas no mundo real. (ARKIN, 1998).

A.2.5 Arquitetura de colônia

A arquitetura de colônia (CONNELL, 1989) é descendente direto da arquitetura de subsunção, utilizando estratégias simples de coordenação, como a supressão como estratégia de coordenação e a especificação das relações entre os comportamentos de maneira mais flexível. A prioridade dos comportamentos na arquitetura de colônia é definida na forma de árvore ao invés de camadas como é feita na arquitetura subsumption (ARKIN, 1998).

A.3 Arquitetura Híbrida

As arquiteturas híbridas buscam combinar os dois tipos de arquiteturas descritas anteriormente, utilizando o melhor de cada uma. Esse tipo de arquitetura visa utilizar o planejamento das arquiteturas deliberativas e o tempo de resposta às mudanças no ambiente das arquiteturas reativas.

Assim, a arquitetura híbrida procura ser adequada para solução de problemas complexos atingindo objetivos de maneira ótima e eficiente (utilizando deliberação) em ambientes dinâmicos que exigem rapidez na resposta (utilizando reatividade). De forma que, a camada deliberativa é utilizada para planejar as ações do robô a partir de uma representação interna global do conhecimento do mundo, de forma a atingir os objetivos do robô com eficiência; e, a camada reativa recebe os planos efetuados pela camada deliberativa e os executa utilizando suas propriedades reativas, de forma a responder rapidamente a mudanças dinâmicas no ambiente.

As arquiteturas híbridas podem ser classificadas dependendo da maneira que o planejamento é realizado, sendo assim subdivida em quatro classificações (ARKIN, 1998): Seleção, Aconselhamento, Adaptação e Adiamento.

A.3.1 Seleção

O planejamento é visto como configuração da camada reativa. O componente de planejamento determina a composição de comportamento e os parâmetros utilizados durante a execução. O planejamento reconfigura a parte reativa de acordo com a percepção. Exemplos: AuRA, DAMN,SFX.

A.3.2 Aconselhamento

O planejamento é visto como um aconselhamento. O planejador sugere mudanças que sistema reativo pode ou não usar. Exemplos: ATLANTIS, 3T.

A.3.3 Adaptação

O planejamento é visto como adaptação. O planejador altera constantemente o componente reativo de acordo com a mudança da percepção do mundo da camada reativa. Exemplos: Planner-Reactor, SSS, DD&P.

A.3.4 Adiamento

O planejamento é visto como um processo pelo compromisso. O planejador adia enquanto for possível as decisões sobre novas ações. Isso permite que os recentes dados do sensor, ao adiar as ações reativas até absolutamente necessário, para fornecer um curso de ação mais eficaz do que seria desenvolvido se um primeiro plano fosse gerado desde o início. Nesta arquitetura espera-se que o componente reativo possa resolver o problema sem intervenções, de forma que os planos são elaborados apenas quando necessário. Exemplos: PRS, Saphira, Arquitetura de Agente.

A.4 AuRA

A arquitetura AuRA é considerada a arquitetura híbrida mais antiga, e foi desenvolvida por Arkin na década de 80, que se baseou na teoria de esquemas. Esta arquitetura subdivide o sistema de navegação em cinco subsistemas: Aprendizagem, Interface com o usuário, Planejador, Cartógrafo e Controlador de esquemas; como mostrado na Figura A.8.

Dos subsistemas, o Controlador de esquemas compõe a parte reativa do sistema, enquanto os subsistemas Planejador e Cartógrafo compõem a parte deliberativa. O Planejador é responsável pela missão e planejamento de tarefas, sendo subdividido em três componentes, equivalentemente à arquitetura Nested Hierarchical Controller (MURPHY, 2000).

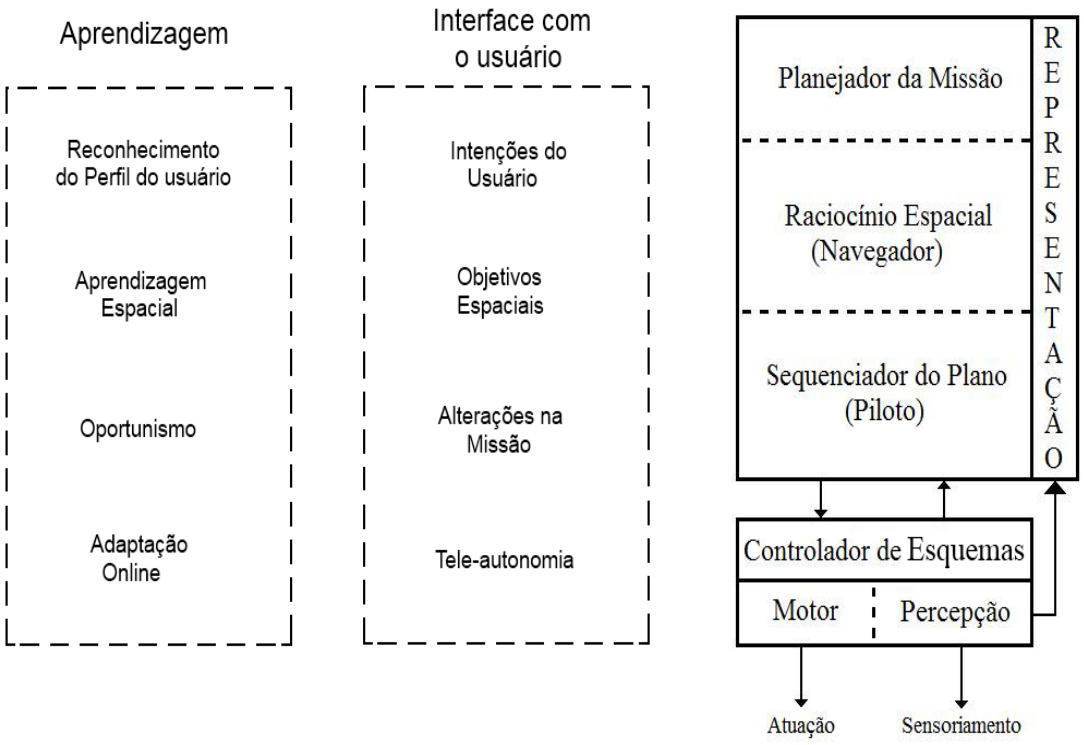


Figura A.8 – Arquitetura AuRA.

O uso de Motor Schemas (comportamentos), ao invés de pré-determinar uma rota exata e tentar coagir o robô a segui-la, permite que o robô se adapte às mudanças no ambiente com sucesso, enquanto ainda satisfaz os objetivos de nível superior.

As vantagens deste tipo de navegação incluem o cálculo rápido e a capacidade de ser mapeados em arquiteturas paralelas, tornando a resposta em tempo real facilmente atingível. A modularidade da arquitetura proporciona facilidade na adequação dos comportamentos e a integração de novos comportamentos, simplificando tanto na manutenção do sistema quanto na facilidade de transferência para novos ambientes e outros tipos de problemas.

O esquema de funcionamento da arquitetura AuRA é mostrado na Figura A.8. Abaixo foi detalhada cada algumas das partes que o compõe (ARKIN, 1998):

a) Planejador da Missão: Componente utilizado para interface com os usuários. Responsável por estabelecer objetivos de alto-nível para o robô e restrições dentro das quais o robô deve operar.
b) Raciocínio Espacial: É o navegador, utiliza o conhecimento cartográfico do ambiente, armazenado em uma memória de longa duração, para construir uma seqüência de trechos de navegação que devem ser executadas pelo robô para que a missão possa ser completada.
c) Seqüenciador de Plano: É o piloto, para cada trecho de navegação gerado pelo módulo de raciocínio espacial, o seqüenciador de plano especifica um conjunto de comportamentos que devem ser ativados para execução. Originalmente, o seqüenciador de plano era um sistema baseado em regras. Mais recentemente ele foi implementado como um seqüenciador de estados finitos (MACKENZIE; CAMERON; ARKIN, 1995).
d) Controlador de Esquemas: Responsável pelo controle e monitoramento dos comportamentos reativos em tempo de execução.

A navegação utilizada na arquitetura AuRA é baseada em comportamentos, onde cada comportamento gera um vetor resposta de modo análogo ao método de campos potenciais. Os esquemas operam de forma assíncrona transmitindo seus resultados para um módulo que soma e normaliza estas entradas e transmite o resultado para um sistema de controle de baixo nível responsável pela controle dos motores (ARKIN, 1998).

Uma vez que a parte reativa começa a ser executada, a parte deliberativa não é reativada a menos que seja detectada uma falha na parte reativa, que tipicamente é indicada por uma falta de progresso ou excesso de tempo para a execução da tarefa. Neste ponto, o planejador deliberativo é reativado, começando da camada inferior, reativando uma camada por vez até que o problema seja resolvido. Portanto, após detectada a falha no componente reativo, o seqüenciador do plano é ativado para que forneça uma nova configuração de comportamentos. Esta configuração é determinada pela informação local sobre o ambiente armazenada em uma memória de curto prazo. Se isso não for suficiente, ou seja, a rota está completamente obstruída dentro do contexto local, o módulo de raciocínio espacial é

ativado e tenta estabelecer um novo caminho que desvie da região onde está ocorrendo o problema. Se ainda assim isso não se mostrar satisfatório, o planejador de missão é chamado e o usuário é informado da dificuldade, podendo desistir da missão ou reformulá-la (GRASSI JR., 2006).

A arquitetura AuRA é altamente modular, flexível e geral, de modo que seus componentes podem ser substituídos de acordo com a aplicação e evolução das tecnologias utilizadas para implementar cada um desses módulos. Outra característica é a flexibilidade que ela fornece por intermédio da introdução de métodos adaptativos e de aprendizado. A adaptação é feita alterando-se a importância ou peso dado a cada um dos esquemas motores utilizados na realização da tarefa. Não obstante, a AuRA cobre uma vasta quantidade de problemas, incluindo ambientes como (PIERI, 2002):

- Ambientes de manufatura;
- Navegação tridimensional como as encontradas em domínios aéreos e aquáticos;
- Navegação em ambientes abertos e fechados;
- Competições de robôs;
- Cenários militares;
- Manipulação móvel;
- Equipes de robôs;

A.5 Conclusão

Atualmente, há diversas arquiteturas reativas, deliberativas e híbridas, contudo, nos últimos anos, as arquiteturas híbridas têm sido mais utilizadas devido a seu equilíbrio entre o planejamento e a reação ao ambiente. Como resultado da análise feita neste capítulo a arquitetura AuRA mostrou-se como a melhor opção devido ao balanço entre as partes reativa e deliberativa, utilizando um método baseado na psique humana para acoplar essas partes. Além disso, a generalidade e modularidade da AuRA permitem um futuro

aperfeiçoamento do sistema desenvolvido, sempre mantendo-se atualizada com as tecnologias do período.

ANEXO B – Sistemas de Localização

Um assunto importante na navegação de robôs móveis é a tarefa de localização, pois um posicionamento correto é uma condição básica para efetuar-se uma navegação efetiva de forma a atingir todos os objetivos, já que sem o conhecimento de sua posição é improvável realizar um deslocamento complexo, como sair de casa e ir ao supermercado.

A tarefa de localização consiste em utilizar um mapa (representação interna do mundo) para encontrar sua posição através da percepção do ambiente ao seu redor. Esta tarefa possui dois grandes obstáculos, a correta atualização do mapa e a presença de erros e/ou ruídos nas leituras dos sensores, que podem interferir decisivamente na realização da tarefa de localizar-se em um ambiente.

Os erros e/ou ruídos presentes na medição, apesar de geralmente pequenos, podem causar grande interferência no deslocamento entre um ponto e outro, pois embora o erro possa ser mínimo, este desvio pode ocasionar numa grande diferença de posicionamento num período maior.

Uma atualização incorreta do mapa interno pode levar o robô a cometer erros de planejamento, induzindo-o a percorrer longos caminhos ou ainda a esbarrar num ponto sem saída devendo retornar ao ponto inicial, ou ainda perder-se. Este problema representa uma grande dificuldade já que o mundo real é um ambiente dinâmico e constantemente se modifica. Sendo necessário incluir no robô a habilidade de resolver este problema, quando este navegar num ambiente dinâmico.

Para abordar este problema o anexo será dividido em duas partes: Métodos de representação interna do mundo e Métodos de localização.

B.1 Métodos de representação interna do mundo

A forma de representar um ambiente é a utilização de um mapa, onde a escolha correta da representação interna do ambiente é um dos principais fatores que influencia o sucesso da localização do robô. O mapa do ambiente deve ser capaz de armazenar informações suficientes para a localização do robô, porém, a quantidade de informação não deve ser muito grande a ponto de comprometer o desempenho do sistema de controle. A escolha da forma de representação a ser utilizada depende da precisão e da simplicidade computacional desejada.

Um mapa apropriado para um robô autônomo precisa relacionar os tipos de dados coletados pelos sensores do robô com a estrutura do ambiente em que esteja navegando (HEINEN, 2002).

Atualmente existem duas principais formas do representação de ambiente utilizadas pelas técnicas de localização: os mapas geométricos e os mapas topológicos.

Os mapas geométricos representam os objetos de acordo com a sua posição geométrica absoluta. Este tipo de mapa pode ser poligonal, onde os obstáculos são representados por polígonos ou baseado em grades (o valor de cada célula indica a presença de um obstáculo naquela posição) (SIEGWART e NOURBAKHSH, 2004).

Os mapas topológicos são baseados nas relações geométricas observadas entre as características do ambiente, ao invés de sua posição absoluta. Representam informações de conexões, tipicamente em forma de grafos, onde os nós são os pontos de referência (ou características) e os arcos indicam a relação entre estes pontos e como o robô pode se mover entre eles (SIEGWART e NOURBAKHSH, 2004).

Cada uma destas representações admite várias especificações e descrições específicas de acordo com cada ambiente. Além disto, muitas representações reais possuem ambos, componentes topológicos e métricos (THRUN, FOX e BURGARD, 1998).

No sentido de se explorar as vantagens das representações métricas e topológicas, pode ser apropriado considerar-se a construção de uma representação utilizando-se observações de outras representações menos abstratas. Adicionalmente, é natural se considerar as descrições locais antes de inter-relações em larga escala. Isto conduz naturalmente às camadas hierárquicas de representações sucessivas dos dados do mapa, como as cinco apresentadas a seguir (DUDEK e JENKIN, 1996):

a) Sensorial: Sinais de dados não processados ou transformações sinal-domínio destes sinais.
b) Geométrica: Objetos 2D ou 3D inferidos a partir dos dados dos sensores.
c) Relacional local: Relações funcionais, estruturais ou semânticas entre objetos geométricos que estão próximos uns aos outros.
d) Topológica: As ligações relacionais em larga escala que conectam objetos e localizações através do ambiente com um todo.
e) Semântica: Rótulos funcionais associados com os componentes do mapa.

A geração de mapa é uma necessidade para muitas aplicações nos sistemas robóticos móveis, pois diversos ambientes ocupados por pessoas passam por mudanças constantes. Desta forma, para os robôs ser considerados autônomos devem estar aptos a adaptar a representação interna do mundo às mudanças no ambiente (HEINEN, 2002).

B.1.1 Mapas geométricos

Em termos de robótica móvel, a forma mais explícita de mapeamento é o mapa geométrico no espaço de configuração, cujas possibilidades de movimentações (exceto aquelas regidas pelas restrições não holonômicas,) são dadas explicitamente.

Os mapas geométricos são descrições métricas do ambientes, sejam estas descrições realizadas por linhas, polígonos ou células, respeitando as dimensões, as relações de tamanho e a distância entre objetos e elementos presentes no ambiente. Há dois métodos de representação geométrica: a representação contínua ou a representação discretizada.

REPRESENTAÇÃO CONTÍNUA

A representação contínua é um método para uma decomposição precisa do ambiente, onde a posição de cada característica percebida no ambiente é precisamente marcada no modelo interno de mundo. As duas técnicas utilizadas para uma representação geométrica contínua do ambiente são empregadas através do uso de polígonos ou linhas finitas. Tanto os polígonos quanto as linhas são empregados para mostrar os obstáculos. A Figura B.1 ilustra um exemplo de representação por linhas e um com representação por polígonos.

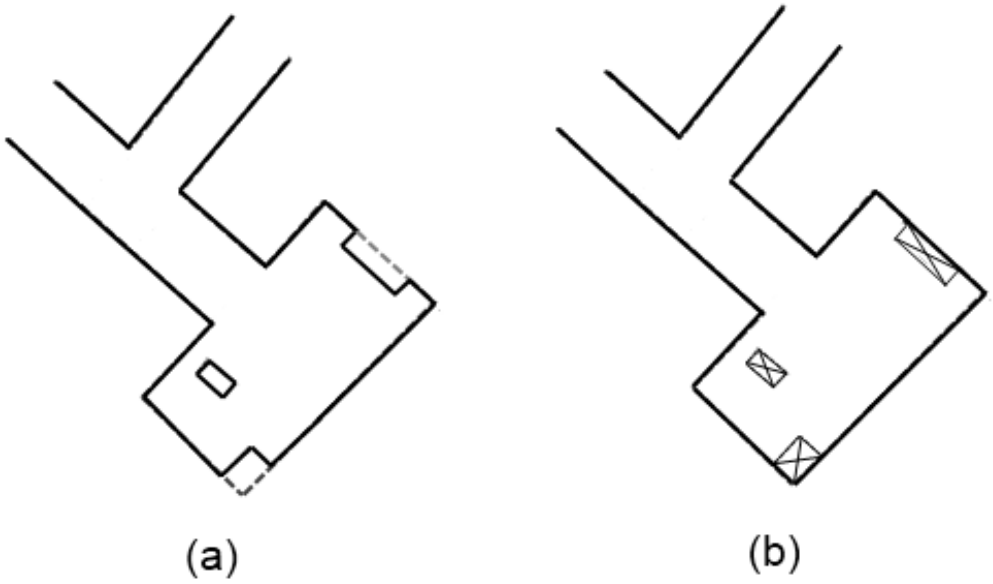


Figura B.1 – Representação contínua do ambiente. (a) representação por linhas; (b) representação por polígonos.

A grande vantagem da representação contínua é a alta precisão e nitidez com respeito ao ambiente. Porém, essa precisão torna esta representação custosa computacionalmente, já que o número de informações armazenadas é muito grande, mas, este problema pode ser resolvido analisando apenas as informações mais relevantes do ambiente, tornando esta técnica, às vezes, inclusive menos custosa.

REPRESENTAÇÃO DISCRETIZADA

A representação discretizada é um método de simplificação do ambiente, de forma a tornar as operações mais rápidas, porém, essa simplificação pode causar perdas de fidelidade, ocultando passagens, ou ainda, abstraindo informações importantes sobre o ambiente.

Uma técnica muito conhecida de discretização é a divisão por células (grids). A técnica de decomposição em células, como o nome já diz, divide o espaço de trabalho em células, pretendendo minimizar a busca pelo caminho mais curto. Existem três métodos que utilizam esta técnica: a decomposição em células fixas, a decomposição em células exatas e a decomposição em células adaptativas (Figura B2).

Em 1991, Latombe (1991) introduz a decomposição em células exatas, ilustrada na Figura B.2b, que diminui significativamente o campo de procura do algoritmo, porém, esta redução no tempo de busca também produz certo grau de imprecisão.

Para minimizar a imprecisão obtida na decomposição em células exatas, uma alternativa é utilizar a decomposição em células fixas, mostrada na Figura B.2c. Essa técnica aumenta o número em que a busca será realizada, aumentado consideravelmente a precisão da trajetória em comparação à decomposição em células exatas. Porém, essa técnica se depara com o mesmo dilema inicial, simplicidade computacional ou precisão, pois quanto menor a célula, maior será a precisão, mas maior será o tempo da busca.

A técnica de decomposição em células adaptativas (ou Octrees e Quadtrees), mostrada na Figura B.2d, cria células de diversos tamanhos, quanto maior o espaço vazio ao redor, maior será a célula. Nesta técnica, a precisão diminui em lugares vazios e aumenta em lugares com mais objetos, e consecutivamente o tempo de busca varia de acordo com o número de objetos no local.

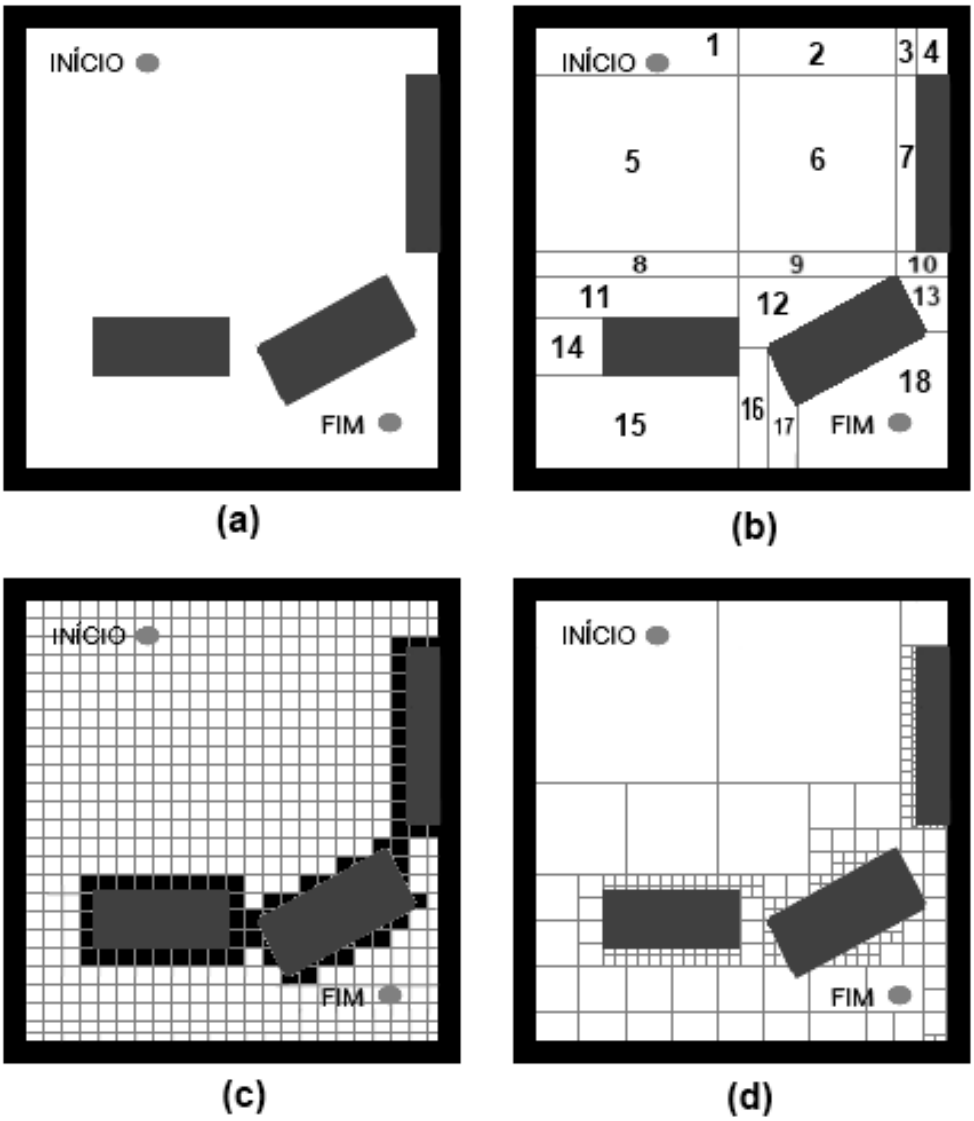


Figura B.2 – Decomposição do ambiente em células. (a) Ambiente original; (b) Decomposição em células exatas; (c) Decomposição em células fixas; Decomposição em células adaptativas.

B.1.2 Mapas topológicos

Contrariamente aos mapas geográficos, a precisão e a escala não são parâmetros relevantes na construção de mapas topológicos, visto que, os mapas topológicos capturam informações qualitativas e rotacionais, descartando informações irrelevantes ou que apresentem detalhes confusos, como resultado, estes têm conexões muito explícitas com as tarefas e as semânticas do problema (HEINEN, 2002).

Um mapa topológico é a descrição de um ambiente através de uma coleção de lugares e caminhos que interligam estes lugares, um exemplo é mostrado na Figura B.3. Os mapas topológicos têm analogias com as percepções espaciais humanas (PARK e KENDER, 1995), por exemplo, ao pedir a informação para alguma pessoa de como chegar a um lugar X, geralmente a resposta seria algo como: ande até o mercadinho, vire à direita, vá até o semáforo, vire à esquerda, etc...").

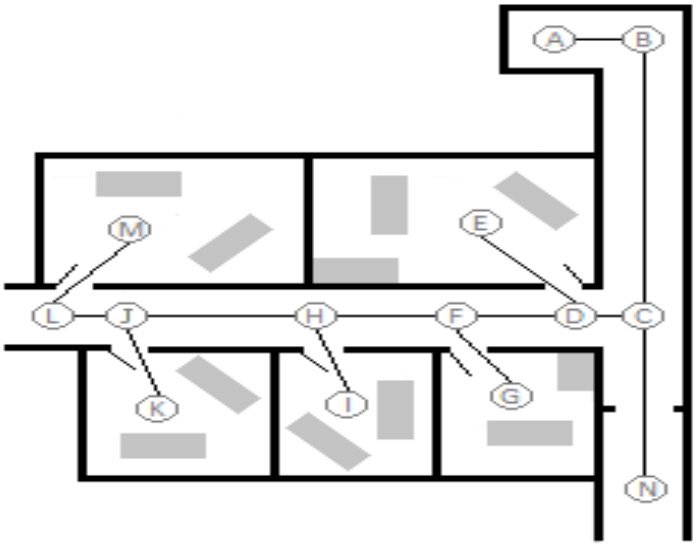


Figura B.3 – Mapa topológico de um ambiente.

Os mapas topológicos dividem o espaço de busca em nós e trajetos, esta representação reproduz a conectividade entre espaços, em outras palavras, essa técnica procura todos os espaços livres e tenta conectá-los entre si (SIEGWART e NOURBAKHSH, 2004). Por exemplo, considerando a Figura B.3, os nós são todas as salas do escritório e cada local que conecta o corredor às salas, e, o caminho que leva de um nó a outros é chamado de trajeto. Assim, um mapa topológico baseado em gráficos pode ser representado por:

$$G = (N; E) \tag{B.1}$$

Onde N é um conjunto de n nós, e E é um conjunto de m trajetos (edge) que conectam os nós. Desta forma, os conjuntos N e E são denotados como:

$$N = \{v_1, v_2, ..., v_n\} \quad \text{(B.2)}$$

$$E = \{e_{12}, e_{13}, e_{23}, ...\} \quad \text{(B.3)}$$

Onde v_i são cada um dos nós existentes no ambiente; e e_{ij} é o trajeto entre o nó i e o nó j.

Um mapa topológico, na sua forma mais pura, não contém nenhum dado métrico, porém, geralmente existem relações métricas no gráfico G, pois os trajetos podem possuir dimensão e orientação com respeito aos nós.

Os mapas topológicos podem ser construídos sem a necessidade de estimar a posição exata do robô, permitindo a integração de grandes mapas, minimizando o problema de erros sensoriais, pois as conexões entre os nós são relativas. Além de desconsiderarem os erros das medições métricas dos sensores, a utilização de mapas topológicos evita o custo de um amplo armazenamento de dados como nas representações métricas. Outra vantagem é devido ao processo de criação de mapas de experiência, pois a consistência dos dados define se o mapa foi gerado corretamente ou não (DUDEK, FREEDMAN e HADJRES, 1993; REMOLINA e KUIPERS, 2004). Entretanto, o maior inconveniente neste tipo de representação é que o sistema deve possuir a capacidade de diferenciar o ponto de referência de um local de outro ponto.

B.2 Métodos de localização

A localização do robô é a análise de um provável local em que as informações recebidas pelo sensoriamento coincidem com o modelo interno do mundo, determinando a posição do robô. O problema da localização é fundamental na robótica móvel autônoma devido a sua importância no planejamento e execução das trajetórias. A competência da localização não se restringe somente a ajudar na navegação, mas também serve para a exploração e para a adaptação do mapa interno do ambiente realizada pelo robô. Portanto, sem um método confiável e preciso de localização, o sistema de navegação encontrará

alguns problemas, como integrar múltiplas leituras sensoriais de um mesmo objeto, inserir ou retira novas características, pontos de referência e obstáculos no modelo interno do mundo, e, o maior inconveniente é não poder seguir de modo adequado a trajetória planejada.

Existem diversas abordagens que tratam o problema de manter uma informação precisa sobre a posição de um robô móvel, sendo possível separar estas abordagens em três métodos: Localização por Odometria, Localização por Marcadores e Localização por Mapas Probabilísticos.

B2.1 Localização por Odometria

A maioria dos seres humanos usa a visão para se localizar, de forma que, vendadas, essas pessoas perderiam integralmente sua noção de localização e da posição dos objetos ao seu redor, nos primeiros passos ela ainda estará com uma certeza alta sobre a sua localização e a dos objetos do local, mas quanto mais ela se deslocar, maior será a sua incerteza. Entretanto, as pessoas com problemas visuais se adaptam para lidar com situações como essa, pois desenvolvem uma alta capacidade de julgar as distâncias percorridas (HEINEN, 2002).

Em robótica móvel, a capacidade de se localizar através da informação das distâncias percorridas é conhecida como *dead reckoning* ou localização baseada em odometria. A apuração das distâncias percorridas é realizada através da observação dos movimentos das rodas utilizando *encoders*, então, através da informação da quantidade de giros que as rodas efetuaram, um robô pode determinar o quanto ele se locomoveu e o quanto ele rotacionou (HEINEN, 2002).

Entretanto, a localização por *dead reckoning* pode ser muito imprecisa devido a derrapagens nas rodas, imprecisão dos sensores, erros de leitura dos sensores e ao fato desta técnica calcular apenas uma aproximação discreta da sua posição, tornando esta técnica perigosa de se utilizada sozinha. Se o robô estiver andando em uma linha reta e os erros

forem pequenos, isto pode ter pouca influência na estimativa de posição, mas se o robô estiver fazendo uma curva, o erro angular pode causar um grande erro translacional, modificando completamente a orientação do robô. Além disso, a freqüência da realização dos cálculos e a precisão dos *encoders* afetam extremamente o desempenho do robô, pois se os erros anteriormente descritos somarem-se com a baixa precisão do *encoder,* ou uma baixa freqüência de verificação de posicionamento, um pequeno erro pode futuramente se transformar num grande desvio, principalmente se este ocorrer durante uma rotação, já que um erro de apenas 1º pode ter muita influência na posição futura de um robô.

B.2.2 LOCALIZAÇÃO POR MARCADORES

Como comentado anteriormente, a maioria dos seres humanos utiliza a visão para se localizar, por exemplo, se alguém pergunta onde fica a farmácia F, a resposta será que fica perto do supermercado S, ou seja, a localização do nosso objetivo, como também a nossa, é estipulada com referência a marcadores, no caso dos humanos, marcadores visuais.

A localização por marcadores baseia-se nos métodos tradicionais de navegação, que utilizam os objetos ao seu redor para se localizar. No caso da robótica móvel, estes marcadores podem ser visuais, sonoros, enviados por ondas eletromagnéticas, ou qualquer outra fonte de energia.

Após estes marcadores serem encontrados, utiliza-se suas posições como referência para determinar a posição atual do robô. Há duas técnicas utilizadas na localização por marcadores: por triangulação e por acumulo de posições.

A técnica da triangulação utiliza o ângulo entre o robô e os marcadores de posição para calcular a posição relativa do robô em relação aos marcadores. Como o nome já refere, é necessário o conhecimento de pelo menos três ângulos para que o cálculo seja realizado, portanto é necessário observar três pontos de referência do ambiente simultaneamente. Um exemplo é o GPS, ilustrado na Figura B.5 (NARA, 2010). Em casos, em que as distâncias entre o robô e os marcadores são conhecidas, dois pontos de referência são suficientes.

A técnica por acúmulo de posições, utiliza o conjunto de informações relativas a posições e direções de um marcador e a quantidade de deslocamento realizado para determinar a localização do robô. Um exemplo é mostrado na Figura B.4b, utilizando marcadores no solo, distantes 50 metros entre cada marcador.

Além disso, o método de localização por marcadores pode ser subdivido em três abordagens principais: localização por marcadores de posição artificiais, localização por marcadores de posição naturais e localização baseada em rotas.

LOCALIZAÇÃO POR MARCADORES DE POSIÇÃO ARTIFICIAIS

A localização por marcadores de posição artificiais consiste em encontrar os objetos inseridos no ambiente, onde cada objeto tem sua posição bem definida no mapa, e assim, a localização do robô pode ser determinada considerando as posições destes marcadores.

Existem dois métodos muito conhecidos de marcadores artificiais utilizados para a localização:

- Landmarks: utiliza objetos artificiais conhecidos no ambiente para determinar a localização do robô, para isto podem ser usados objetos passivos na parede, no chão ou em postes, esses objetos têm características singulares, como cores, formatos ou marcas diferentes. (SIEGWART e NOURBAKHSH, 2004). Dois exemplos são ilustrados na Figura B.4.

- Sistema de Posicionamento por Guias: utilizam guias visuais, sinais ultrasônicos ou sinais eletromagnéticos para determinar a posição do robô (SIEGWART e NOURBAKHSH, 2004);. Um exemplo muito conhecido é o GPS, ilustrado na Figura B.5.

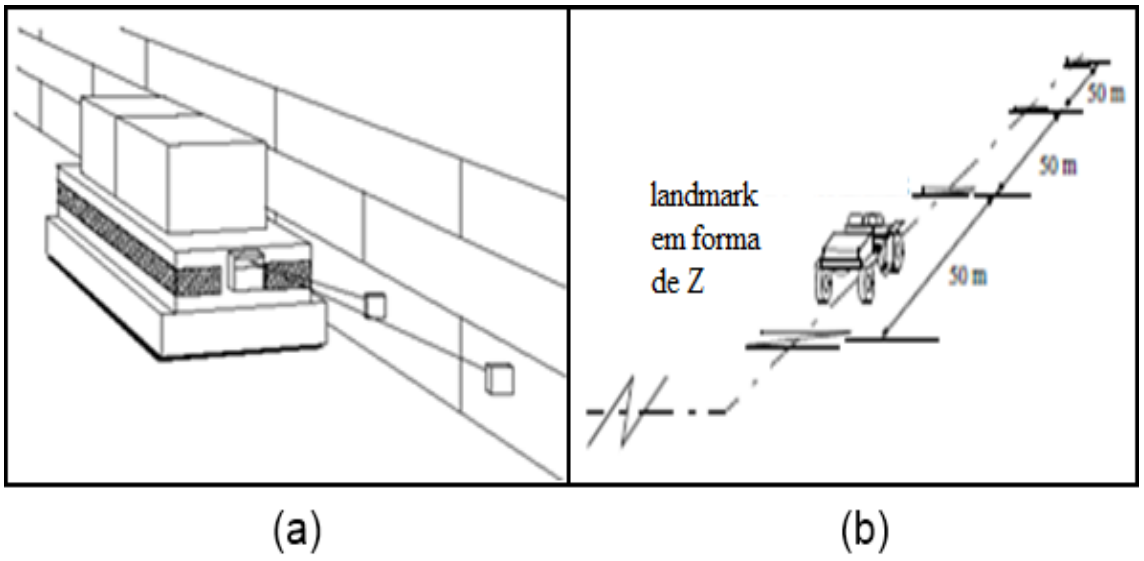


Figura B.4 – Localização por Landmarks (SIEGWART e NOURBAKHSH, 2004). (a) landmarks na parede; (b) landmarks em forma de Z no solo.

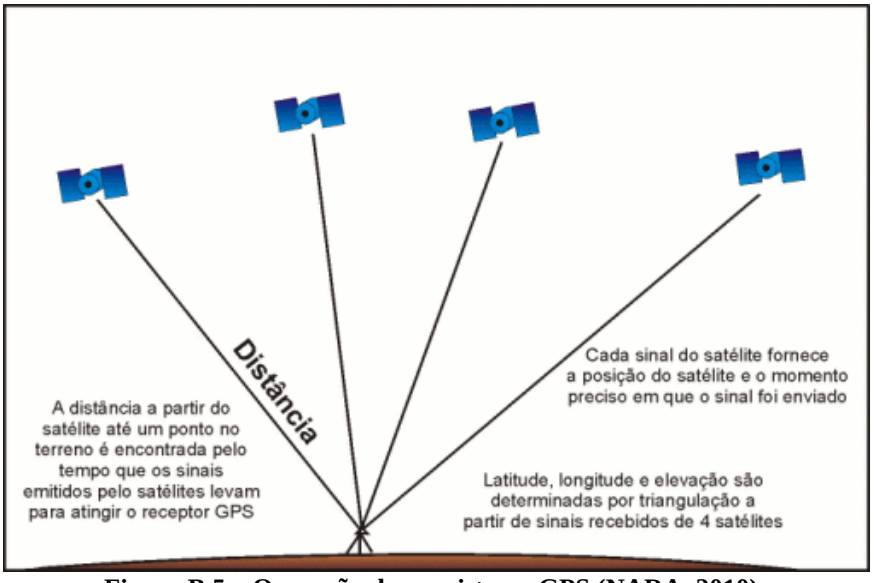


Figura B.5 – Operação de um sistema GPS (NARA, 2010).

Os métodos de localização por marcadores de posição artificiais são muito precisos, porém, possuem uma série de restrições sobre o ambiente e o robô. Pois, neste caso, o ambiente precisa ser modificado, o que nem sempre é possível, e o robô necessita um mapa preciso de todos os pontos de referência do ambiente. É notável que estes métodos possuem grandes inconvenientes para sua aplicação no mundo real.

LOCALIZAÇÃO POR MARCADORES DE POSIÇÃO NATURAIS

A diferença entre a localização por marcadores de posição naturais e a localização do item anterior é a origem dos marcadores utilizados na localização, enquanto os marcadores artificiais são inseridos no ambiente, os marcadores naturais são objetivos já contidos nos ambientes que devem ser distintos dos outros objetos a fim de tornar-se uma referência para a localização. A limitação deste método é a capacidade do robô de distinguir um objeto dos outros.

Para superar esta limitação existem alguns estudos que visam implementar esta capacidade de distinção no robô. Moon, Miura e Shirai (2001) propuseram um método para extrair automaticamente segmentos de linhas verticais distintas em superfícies planares a partir de imagens obtidas por câmeras instaladas em um robô móvel, essas linhas podem ser utilizadas como pontos de referências visuais, tornando possível a aplicação dos métodos de triangulação em ambientes reais (HEINEN, 2002).

Kuipers (1988) descreve uma técnica de localização que utiliza descrições topológicas do ambiente, chamado localização topológica. O trabalho de Kuipers é motivado pelas evidências da ciência cognitiva que indicam que os humanos utilizam informações topológicas, ao invés de métricas, para navegação. O modelo topológico de Kuipers é organizado como um grafo, com os nós representando os pontos de referência (chamados de "distinctive places", ou DPs), e os arcos representando as rotas conectadas. Um DP é definido como um ponto no ambiente que é distinguível localmente de acordo com alguns critérios geométricos, tais como: distância entre objetos, simetria do ambiente, entre outros (HEINEN, 2002).

Existem diversas outras abordagens buscando encontrar a solução para distinguir um objeto dos demais, e todos os métodos assumem que nenhuma posição possui as mesmas características que as demais, consistindo em coletar toda a informação possível ou alguma característica específica de um local e usá-la para determinar a posição atual através de comparação com os dados prévios, portanto, a localização por marcadores de posição

naturais não necessita de modificações no ambiente, porém encontrar estes marcadores no ambiente é extremamente complicado e custoso computacionalmente.

LOCALIZAÇÃO BASEADA EM ROTAS

A localização baseada em rotas é um método muito confiável, pois utiliza a rota demarcada para se localizar, e como está explicitamente marcada, o robô conhece sua posição, entretanto, não é possível determinar exatamente a posição em coordenadas globais, apenas especificar em qual parte da trajetória ele está. Um exemplo, ilustrado na Figura B.6, seria de uma pessoa viajando entre uma cidade A e outra cidade B por uma estrada desconhecida, assim, ela sabe que saiu da cidade A e está indo para a cidade B, mas em que ponto exatamente ela está seria difícil de constatar apenas na avaliação da estrada. (SIEGWART e NOURBAKHSH, 2004)

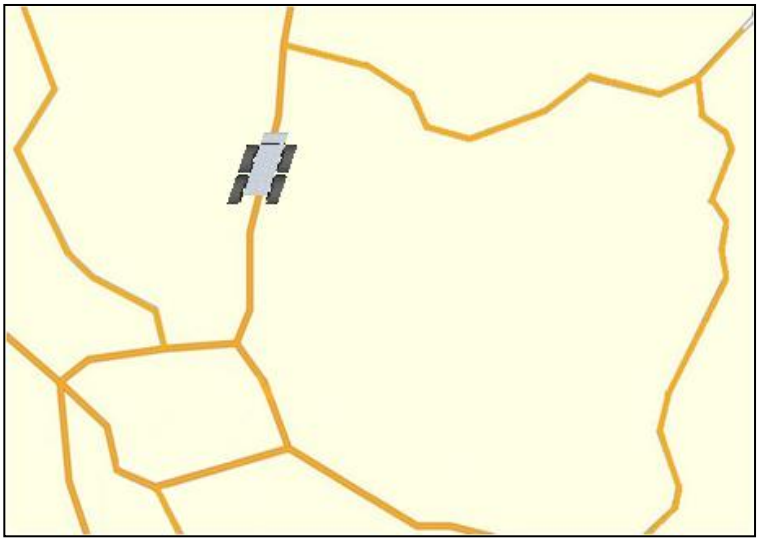


Figura B.6 – Deslocamento entre duas cidades.

Uma abordagem para transpor o problema de localização exata neste método é a utilização a odometria como auxilio de localização. Outra abordagem é utilizar marcas na rota para definição de posicionamento, como o uso de landmarks, tintas transparentes ou radiação ultravioleta.

LOCALIZAÇÃO POR MAPAS PROBABILÍSTICOS

Em muitos casos uma única observação não é suficiente para determinar a posição do robô móvel devido à semelhança de diversos locais. Dessa forma, para resolver com o problema de ambigüidade entre diferentes posições possíveis, é necessário manter estas posições como válidas e coletar informações adicionais em outras posições do ambiente. Em muitos casos basta se locomover em uma direção aleatória até que o número de observações seja suficiente para acabar com a ambigüidade (HEINEN, 2002).

Como percebido, "representação de posição com múltiplas hipóteses é vantajosa porque o robô pode explicitamente controlar suas próprias crenças sobre suas posições possíveis no ambiente". Teoricamente, a crença do robô irá mudar de acordo com a as saídas do motor e as entradas dos sensores até encontrar a localização correta (SIEGWART e NOURBAKHSH, 2004).

Os mapas probabilísticos explicitamente identificam as probabilidades das posições possíveis do robô, e por isso, estão sendo muito pesquisados recentemente (SIEGWART e NOURBAKHSH, 2004). A Figura B.7 ilustra a idéia do funcionamento da localização por mapas probabilísticos. Observa-se na Figura B.7 que num primeiro momento, quando o robô está na frente do primeiro pilar, o robô entende que poderia estar na frente de qualquer pilar existente no mapa. Num momento posterior, quando o robô fica em frente ao segundo pilar, ele se localiza, tendo como condição que ele acabará de passar por um pilar a N metros e encontrou um segundo pilar, e desta forma, a probabilidade de ele estar na frente do segundo pilar é muito alta, tornando-se a localização selecionada.

Os mapas probabilísticos utilizam ferramentas de probabilidades como o Filtro de Bayes, Cadeia de Markov, Monte Carlo, Filtro de Kalman, etc, para calcular a probabilidade de o robô encontrar-se num determinado local. Apesar de haver diversos algoritmos diferentes, existem dois métodos principais: a localização de Markov e a localização com Filro de Kalman.

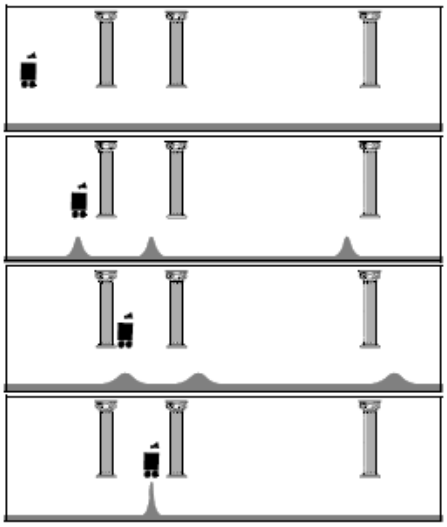


Figura B.7 – Localização por mapas probabilísticos (SIEGWART e NOURBAKHSH, 2004).

A localização de Markov, ao invés de manter uma única hipótese sobre a posição do robô mantém múltiplas hipóteses sobre a posição do robô, onde cada hipótese possui um peso associado calculado matematicamente. A idéia principal é calcular uma distribuição de probabilidades sobre todos os possíveis locais do ambiente (HEINEN, 2002).

O modelo de localização de Markov pode representar qualquer função de densidade de probabilidade sobre a posição do robô, por isso, a localização de Markov permite a localização correta, mesmo começando em qualquer posição, e assim, o robô pode se recuperar de situações ambíguas. No entanto, para atualizar a probabilidade de todas as posições dentro do ambiente é necessária muita memória e poder computacional, podendo ser necessário limitar a precisão e tamanho do mapa (SIEGWART e NOURBAKHSH, 2004).

"O filtro de Kalman é um mecanismo matemático para produzir uma estimativa ótima do estado do sistema com base no conhecimento do sistema e do dispositivo de medição, a descrição do ruído do sistema e erros de medição e as incertezas nos modelos de dinâmica". Portanto, o filtro de Kalman é utilizado de forma otimizada para a fusão de sensores.

A teoria do filtro de Kalman determina que o sistema deva ser assumido linear e com ruído branco gaussiano, de forma que a hipótese de erro de Gauss é inválida para estas aplicações de robôs móveis. Assim, se a incerteza do robô se tornar muito grande, o filtro de Kalman pode falhar. Porém, conhecendo a posição inicial, a utilização do filtro de Kalman produz uma localização precisa e computacionalmente eficiente. (SIEGWART e NOURBAKHSH, 2004)

Outra dificuldade que as técnicas de localização por mapas probabilísticos resolvem de forma determinante é a localização em ambientes dinâmicos. A maioria das técnicas de posicionamento assume que o ambiente é estático para conseguir se localizar, porém, num ambiente do cotidiano humano existem muitos obstáculos móveis. Para evitar as interferências causadas por obstáculos não modelados no ambiente, os mapas probabilísticos filtram as leituras dos sensores, descartando as leituras relacionadas aos obstáculos desconhecidos e/ou móveis. Os dois filtros utilizados principalmente para esta filtragem são: o Filtro de Entropia e o filtro de distância. (HEINEN, 2002).

B.3 Conclusão

O mapeamento e a localização do robô são fundamentais no planejamento e execução de um caminho, pois um pequeno erro pode causar um grande desvio na rota desejada, e para realizar estas tarefas foram criadas diversas técnicas que procuram abordar o problema da localização de formas diferentes. Neste trabalho, considerando as características da plataforma selecionada (sensores seguidores de linha e pequena memória de armazenagem de dados), optou-se pela utilização de um sistema baseado em mapeamento topológico utilizando rotas para localizar o robô. Além disso, a similaridade do mapeamento topológico com o método humano de localizar-se, e a confiabilidade da localização fornecida pela localização por rotas influenciaram muito nas suas escolhas.

ANEXO C – Planejamento de Movimento

O planejamento baseado em um modelo interno do mundo pode ser feito em diferentes níveis de abstração, variando desde um nível mais alto (planejamento de missões e de tarefas) até um nível mais baixo (planejamento de movimento), que procura resolver principalmente o problema de navegação autônoma (LAVALLE, 2006). Neste anexo será abordado o nível mais baixo de planejamento.

O planejamento de movimento tem como função converter os dados do planejador da missão ou do usuário, que é o objetivo, em uma trajetória utilizando um mapa (modelo interno do mundo). Este planejamento pode ser realizado em duas fases distintas: no planejamento de caminhos e no planejamento de trajetórias.

No planejamento de caminhos, o planejador encontra o melhor caminho para atingir o objetivo utilizando o mapa interno do ambiente; e no planejamento de trajetórias, o planejador determina os movimentos que o robô irá realizar para atingir todos os sub-objetivos. Em alguns casos, quando o ambiente em que o robô esta inserido for pequeno, o planejamento de movimento se abstém de utilizar o planejador de caminhos, utilizando apenas o planejamento de trajetórias. Porém, quanto maior o ambiente, menor será o sucesso do planejador de trajetórias ao executar o planejamento. Portanto, num ambiente grande e com muitas restrições (obstáculos, paredes, corredores pequenos), a utilização do planejador de caminhos aumentará extremamente a probabilidade de sucesso. Maiores detalhes e técnicas utilizadas para cada fase serão apresentadas nas partes C.1 e C.2.

C.1 Planejamento de Caminhos

Planejamento é a tarefa de chegar a um objetivo através de uma seqüência de ações (RUSSELL e NORVIG, 2003), portanto, o planejamento de caminhos é a utilização do modelo interno do mundo para a determinação dos sub-objetivos a serem alcançados pelo robô. No caso de robôs móveis, são os locais que o robô deve passar para atingir seu objetivo.

Por exemplo, uma pessoa está num estacionamento de um supermercado (Figura C.1), acabou de sair do elevador e quer ir até seu carro (preto) que está na vaga 20. Qual caminho ela deverá seguir? Ela tem muitas opções: passar pela vaga 23, ir para vaga 30, 29, 28 e assim chegar ao seu carro; ir pelo meio da pista e passar entre os pilares B e C; passar pela vaga 32, passar por trás do carro azul da vaga 31, e passar pelas vagas 30, 29, 28 até chegar a seu carro; entre outros. Existem inúmeros caminhos possíveis, mas a pessoa deve escolher qual é o melhor, mais rápido, mais seguro, mais curto, etc., o critério de escolha dependerá da situação.

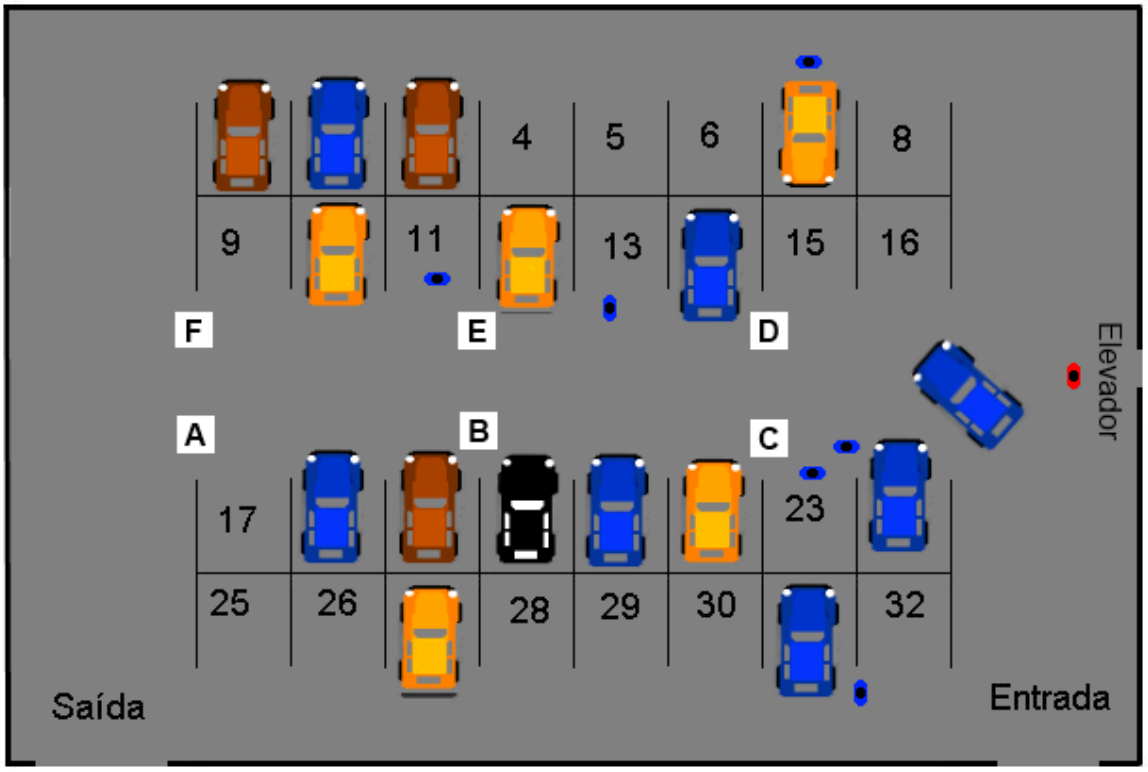


Figura C.1 – Representação de um estacionamento.

Em robótica móvel, um robô assumiria o lugar da pessoa, tendo que determinar o melhor caminho para atingir seu objetivo (o carro laranja). Para realizar esta tarefa existem quatro abordagens conhecidas: Planejamento por mapas de estradas, Decomposição em células, Campos Potenciais e Waypoints. E, existem inúmeros algoritmos de busca para determinar o melhor caminho (CORMEN, 2001). Nas próximas seções serão detalhadas as quatro abordagens, e após, serão apresentados alguns algoritmos de busca.

C.1.1 Planejamento por Mapas de Estradas

A abordagem utilizando mapas de estradas consiste em capturar a conectividade do espaço livre do ambiente em uma rede de curvas de uma dimensão, ou seja, esta estratégia procura estabelecer diversas rotas entre o ponto inicial e o ponto final (Figura C.2), e então, escolhe a rota mais curta (SIEGWART e NOURBAKHSH, 2004).

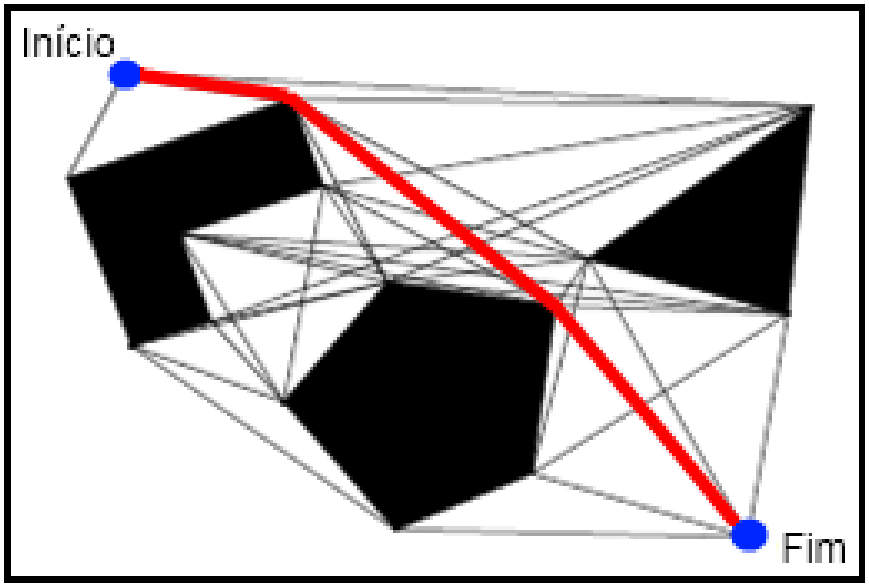


Figura C.2 – Mapas de estradas com o gráfico de visibilidade (SIEGWART e NOURBAKHSH, 2004)

O planejamento por mapas de estradas é uma decomposição do espaço de trabalho do robô baseado especificamente na geometria dos obstáculos. O desafio é construir um conjunto de estradas que, juntas, permitam que o robô possa ir a qualquer lugar, minimizando o número total de estradas.

Para isso são utilizados principalmente dois métodos: os gráficos de visibilidade e os diagramas de Voronoi. No caso do gráfico de visibilidade (Figura C.2), as estradas chegam o mais perto possível de obstáculos e os caminhos resultantes são soluções de comprimento mínimo. No caso do diagrama de Voronoi (Figura C.3), as estradas ficam no meio da passagem entre os obstáculos, obtendo caminhos mais longos que os encontrados no gráfico de visibilidade, porém mais seguros.

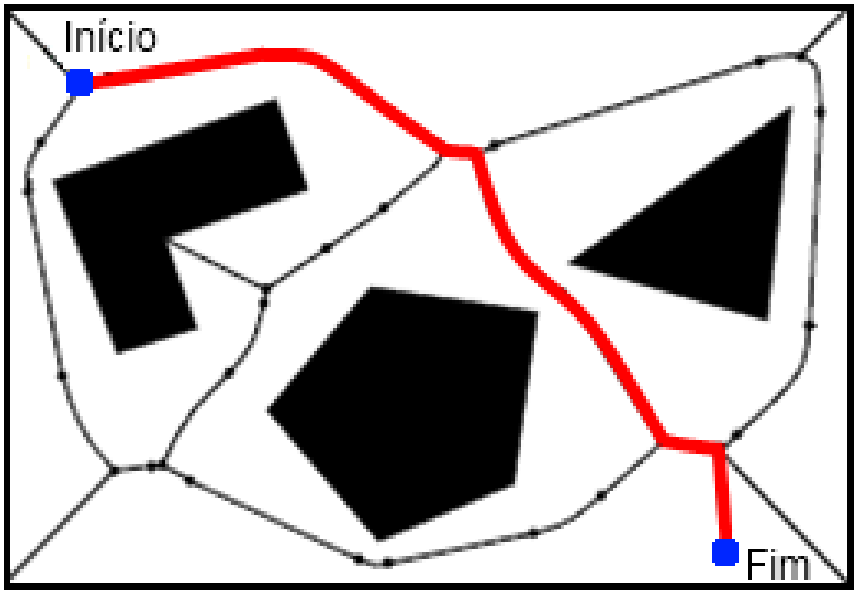


Figura C.3 – Mapas de estradas com diagrama de Voronoi (SIEGWART e NOURBAKHSH, 2004)

C.1.2 Decomposição em Células

A estratégia utilizando Decomposição em células consiste em dividir o espaço livre do robô em regiões simples (células), indicando quais delas estão livres ou ocupadas por um objeto. E, a partir deste mapa, o planejador encontrar um caminho entre o ponto inicial e o ponto final através de células livres adjacentes (SIEGWART e NOURBAKHSH, 2004). Um exemplo é a Figura C.4, onde o robô deve se deslocar entre os dois pontos em vermelho, e para isso, planeja seguir o caminho mostrado pelo em azul, e assim, definindo as células que compõe o caminho até o objetivo.

Cada tipo de decomposição possui um método diferente de busca, o que difere o tempo de busca e a precisão. O método mais utilizado entre os usuários do planejamento por decomposição em células é o de células fixas, pois como as células sempre possuem as mesmas dimensões, pode-se controlar a precisão e o tempo computacional.(SIEGWART e NOURBAKHSH, 2004)

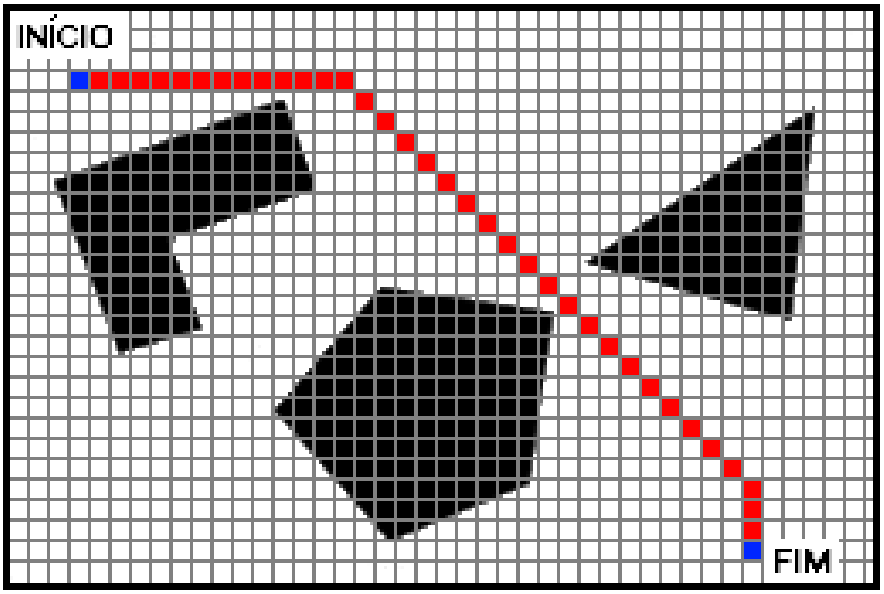


Figura C.4 – Planejamento por decomposição em células.

C.1.3 Campos Potenciais

A estratégia utilizando Campos Potenciais consiste em associar um valor a cada posição do mapa segundo uma função análoga ao campo potencial entre cargas elétricas, de modo que o objetivo seja uma força atrativa e um obstáculo seja uma força repulsiva (SIEGWART e NOURBAKHSH, 2004).

O mapa é criado de forma que o objetivo seja o mínimo ponto, habilitando o robô a encontrar um caminho onde o robô alcance seu objetivo com o menor consumo de energia. Este método é mais que um planejador de caminhos, ele atua como um planejador de movimentos e um controlador, pois atua de maneira que o robô conheça qual é a próxima direção que deve tomar devido às forças virtuais criadas no mapa. Um exemplo é mostrado na Figura C.5, onde o ponto inicial é a região em vermelha mais alta, o ponto objetivo é a região mais baixa e os obstáculos são representados pelos picos, pois têm um valor potencial maior que os locais ao redor, assim, o robô deve se mover para a região de menor valor potencial, seria como largar um esfera do ponto mais alto (ponto inicial) de um campo com este formato, onde a esfera desceria até para na parte mais baixa (o objetivo) (SIEGWART e NOURBAKHSH, 2004).

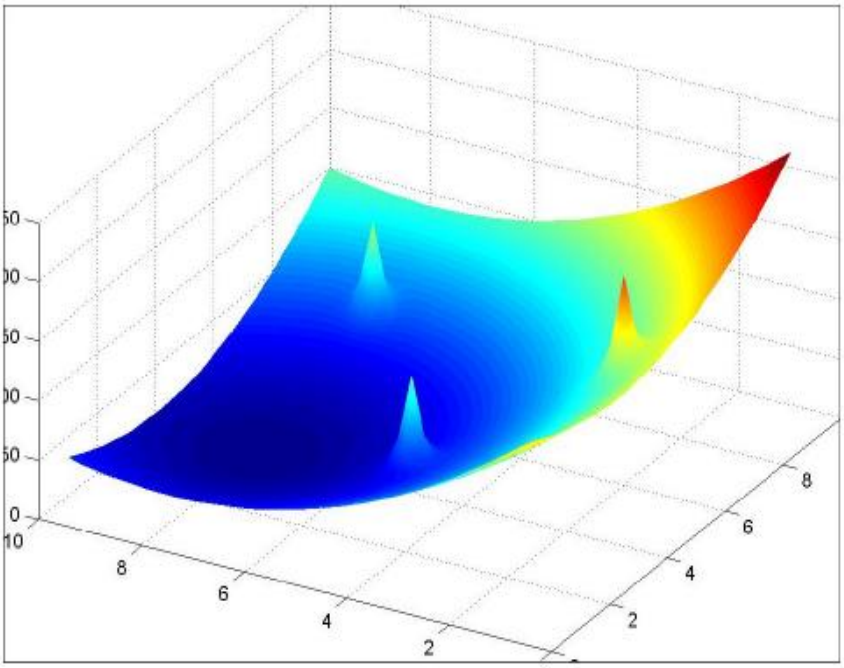


Figura C.5 – Planejamento através de campos potenciais (SIEGWART e NOURBAKHSH, 2004).

C1.3 Waypoints

Waypoints são utilizados para navegação desde que os humanos começaram a navegar, sendo associados a lugares com características distintas, por exemplo, rios, formações rochosas e quedas de água. Portanto, trajetos muito longos de navegação eram divididos em trechos menores, caracterizados como o deslocamento de um waypoint a outro até atingir o objetivo final. Por exemplo, sabia-se que para chegar a cidadela G era necessário ir até a montanha Y, dobrar à esquerda, atravessar o rio pela ponte de madeira, ir até a floresta, dobrar à direita e andar até chegar a cidadela G.

Atualmente, estas associações persistem, mas outros artifícios começaram a serem mais utilizados para pontos de controle, como faróis, bóias e satélite para navegação. Além disso, com a popularização do GPS, os Waypoints começaram a ser associados a um conjunto de coordenadas, geralmente, usando a latitude e a longitude para navegação terrestre, e, para navegação aérea, também usa-se a altitude. A Figura C.6 mostra um percurso entre os pontos A e B dividido em 6 trechos utilizando waypoints.

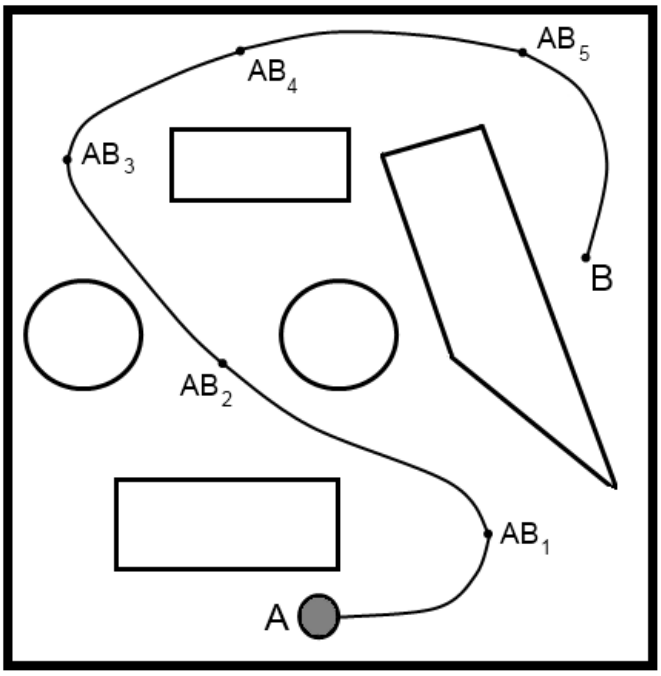


Figura C.6 – Trecho entre A e B subdividido em 6 waypoints.

C1.4 Algoritmos de Busca

As abordagens citadas anteriormente são utilizadas para dividir o ambiente em nós ou grids, diminuindo os locais a serem analisados, tornando a busca pelo melhor caminho mais rápida.

Depois que o ambiente é divido, um algoritmo de busca analisa todos os pontos para descobrir por quais o robô deverá passar para atingir o objetivo. Atualmente, existem diversos algoritmos de busca, pomo os algoritmos Best-first search, Breadth-first search, Depth-first search, Greedy search, A*, D*, Hill climbing e o algoritmo de Dijkstra (CORMEN, 2001).

Breadth-first search é um algoritmo de procura em grafos que procura encontrar as distâncias entre um nó inicial e cada nó atingível num grafo G = (N, E). Este algoritmo explora todos os nós vizinhos do nó inicial, depois, explora os nós vizinhos inexplorados destes nós, e continua até encontrar o nó objetivo.

O pseudo-algoritmo BFS (algoritmo C.1) mostrado abaixo assume que a entrada do grafo G é representada usando uma lista de adjacência, mantendo diversos dados adicionais ao nó no grafo. A cor de cada nó $u \in N$ é armazenado numa variável *color[u]*, a distância entre o nó inicial *s* e o nó *u* é armazenado num variável *d[u]*, e o predecessor de *u* é armazenado numa variável $\pi[u]$. A cor de cada nó pode ser branca, cinza ou preta, onde a cor branca representa os nós não visitados, a cor preta representa os nós visitados, e a cor cinza representa os nós descobertos e ainda não visitados.

```
BFS(G, s)
1 for each vertex u ∈ N[G] − {s}
2       do color[u] ← WHITE
3          d[u]←∞
4          π[u]← NIL
5 color[s] ← GRAY
6 d[s] ← 0
7 π[s] ← NIL
8 Q ← ∅
9 ENQUEUE(Q, s)
10 while Q _= ∅
11      do u ← DEQUEUE(Q)
12         for each v ∈ Adj[u]
13              do if color[v] = WHITE
14                 then color[v] ← GRAY
15                      d[v] ← d[u] + 1
16                      π[v]← u
17                      ENQUEUE(Q, v)
18         color[u] ← BLACK
```

Algoritmo C.1 – Pseudo-algoritmo Breadth-first search (CORMEN, 2001).

O algoritmo A* é o mais conhecido entre os algoritmos de busca Best-first search (RUSSELL e NORVIG, 2003). Este algoritmo encontra o caminho com menor custo de um nó inicial para um nó objetivo, não sendo possível a procura de um caminho com múltiplos objetivos.

O algoritmo A* é uma extensão do algoritmo de Dijkstra, alcançando a resposta de forma mais rápida utilizando heurística. Assim, o algoritmo avalia os nós somando g(n), custo para alcançar o nó atual, com h(n), custo pra atingir o nó objetivo a partir do nó atual.

Segundo (RUSSELL e NORVIG, 2003), o A* é um algoritmo que gera um caminho ótimo desde que a função heurística adotada para a estimação da função h(n) seja admissível, nunca superestimando o custo em relação ao custo real. Uma função comumente utilizada para h(n) é a distância euclidiana entre a localização do nó avaliado e a localização do nó objetivo.

Lester em (LESTER, 2005) descreve o conceito do algoritmo A* e apresenta um método para implementar este algoritmo. Esta metodologia é mostrada abaixo.

Resumo do Método A *

1) adicione o quadrado inicial à lista aberta.

2) Repita o seguinte:

    a. Procure o quadrado que tenha o menor custo de F na lista aberta. Nós referimos a isto como o quadrado corrente.

    b. Mova-o para a lista fechada.

    c. Para cada um dos 8 quadrados adjacente a este quadrado corrente.

        i. Se não é passável ou se estiver na lista fechada, ignore. Caso contrário faça o seguinte:

            1. Se não estiver na lista aberta, acrescente-o à lista aberta. Faça o quadrado atual o pai deste quadrado. Grave os custos F, G, e H do quadrado.

            2. Se já estiver na lista aberta, confere para ver se este caminho para aquele quadrado é melhor, usando custo G como medida. Um valor G mais baixo mostra que este é um caminho melhor. Nesse caso, mude o pai do quadrado para o quadrado atual, e recalcule os valores de G e F do quadrado. Se você está mantendo sua lista aberta ordenada por F, você pode precisar reordenar a lista para corresponder a mudança.

d. Pare quando você:

i. Acrescente o quadrado alvo à lista fechada o que determina que o caminho foi achado, ou

ii. Não ache o quadrado alvo, e a lista aberta está vazia. Neste caso, não há nenhum caminho.

3) Salve o caminho. Caminhando para trás do quadrado alvo, vá de cada quadrado a seu quadrado pai até que você alcance o quadrado inicial. Isso é seu caminho.

Algoritmo C.2 – Metodologia para implementação do algoritmo A* (LESTER, 2005).

O algoritmo de Dijkstra foi criado pelo holandês Edsger Dijkstra, e é um algoritmo que resolve o problema de menor caminho a partir de um nó inicial para um grafo com trajetos com custos não-negativos produzindo uma árvore de menor caminho.

O algoritmo de Dijkstra mantém um conjunto S com os nós que já tiveram suas distâncias até o nó inicial determinadas. O algoritmo seleciona um nó $u \in N - S$ que possuir a estimativa mínima de caminho mais curto, descobre todos os trajetos que partem de *u*, e adiciona *u* à *S*, repete este procedimento até que todos os nós sejam verificados. Um pseudo-algoritmo do algoritmo de Dijkstra é mostrado abaixo.

```
DIJKSTRA(G,w, s)
1 INITIALIZE-SINGLE-SOURCE(G, s)
2 S ← ∅
3 Q ← N[G]
4 while Q != ∅
5       do u ← EXTRACT-MIN(Q)
6          S ← S ∪ {u}
7          for each vertex v ∈ Adj[u]
8                do RELAX(u, v,w)
```

Algoritmo C.3 – Pseudo-algoritmo do algoritmo de Dijkstra (CORMEN, 2001).

C.2 Planejamento de Trajetórias

Planejamento de trajetória é por vezes referido, erroneamente, como planejamento de movimento ou ainda como planejamento de caminhos. Planejamento de caminhos é distinto do planejamento de trajetória já que é parametrizado pelo tempo, enquanto, essencialmente, o planejamento trajetória abrange o planejamento de caminhos, e adicionalmente, planeja seus movimentos baseando-se na velocidade, no tempo e na cinemática da plataforma robótica utilizada.

O planejamento da trajetória entre um ponto A e um ponto B é realizado através da computação das velocidades que os atuadores devem possuir para se deslocar entre o ponto A e o ponto B sem colidir com algum objeto. Para isto é necessário considerar o espaço de trabalho livre, a holomicidade da plataforma robótica e a dinamicidade do ambiente.

Os dois métodos mais utilizados para realizar o planejamento de trajetória são a utilização de uma ou mais funções de curvas pré-definidas e a utilização de cálculos analíticos e geométricos.

As funções das curvas podem ser obtidas através de polinômios parametrizados, sendo utilizadas como referência para a movimentação do robô, onde o robô, conhecendo a localização atual, computa a futura localização desejada através da função previamente calculada, como, por exemplo, em (WALTHER, STEINHAUS e DILLMANN, 2010), onde os autores utilizaram β-splines para encontrar a função para percorrer a trajetória desejada.

Uma β-splines é uma função polinomial seccional de certo grau, sendo aconselhável para caminhos suaves e livres de colisão (BERGLUND, JONSSON, e SDERKVIST, 2003). E, sendo as β-splines uma generalização das splines de Bézier, é possível definir a curva utilizando a equação C.1 (WEISSTEIN, 2010).

$$C(t) = \sum_{i=0}^{n} P_i * N_{i,j}(t) \qquad \text{(C.1)}$$

Komoriya e Tanie (1989) foram os primeiros a propor a utilização de β-splines para interpolar waypoints em um caminho (ALVES NETO, 2008), onde apresentaram um método para gerar curvas β-splines de terceira ordem que passam por pontos específicos. Atualmente, splines têm sido muito utilizadas em todos os tipos de robôs, desde em robôs manipuladores (DU PLESSIS e SNYMAN, 2003), em robôs terrestres (CHOI, CURRY e ELKAIM, 2010), e até em robôs aéreos (KOYUNCU e INALHAN, 2008; SHARMA, KULCZYCKI e ELFES, 2007), pois a precisão e suavidade alcançadas são muito grandes, como mostram a Figuras C.7, C.8 e C.9.

Como observado nas Figuras C.7, C.8 e C.9, a trajetória planejada passa pelos pontos esperados com uma curva contínua e suave, contudo, para alcançar um nível satisfatório de precisão é necessário utilizar um grande número de pontos, tornando o processamento muito demorado.

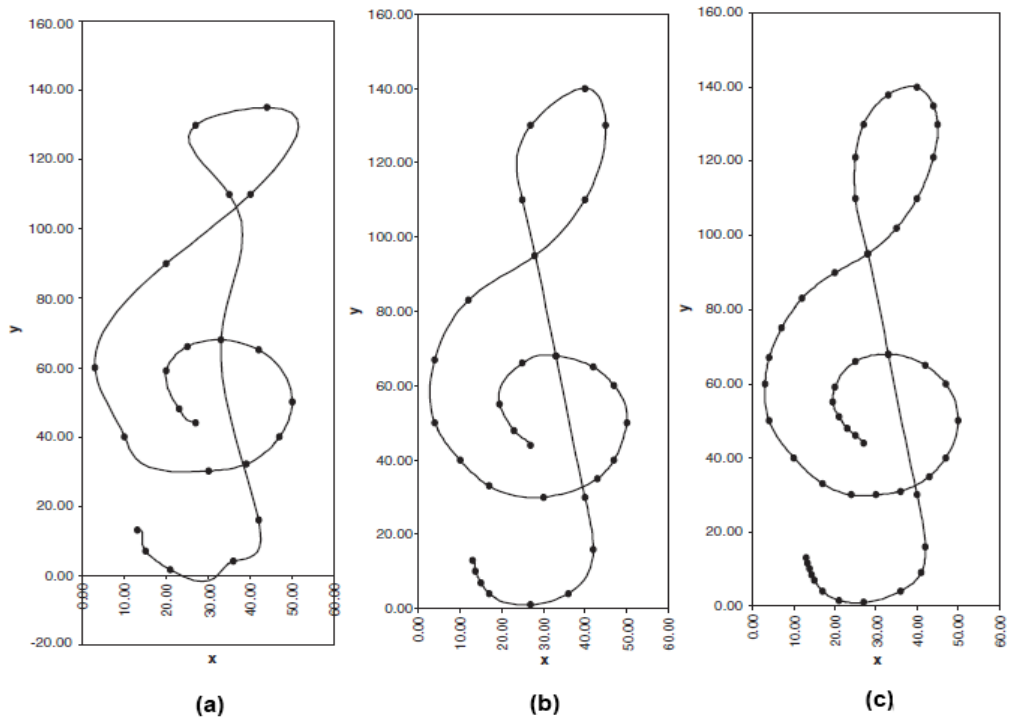


Figura C.7 – Utilização de splines para manipuladores robóticos (DU PLESSIS e SNYMAN, 2003). (a) 22 pontos; (c) 31 pontos; (c) 49 pontos.

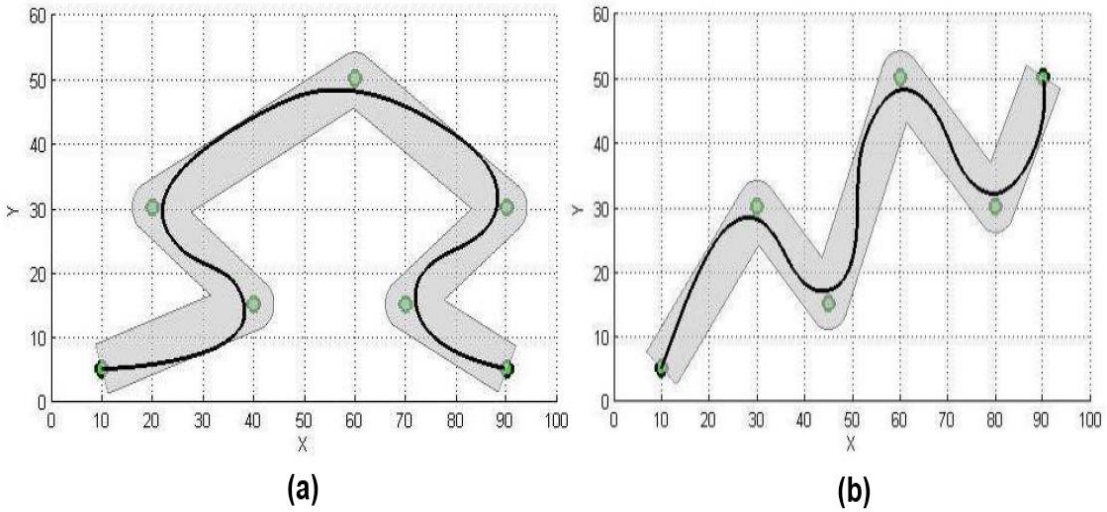


Figura C.8 – Utilização de splines para robôs terrestres (CHOI, CURRY e ELKAIM, 2010). (a) Trajeto 1; (b) Trajeto 2.

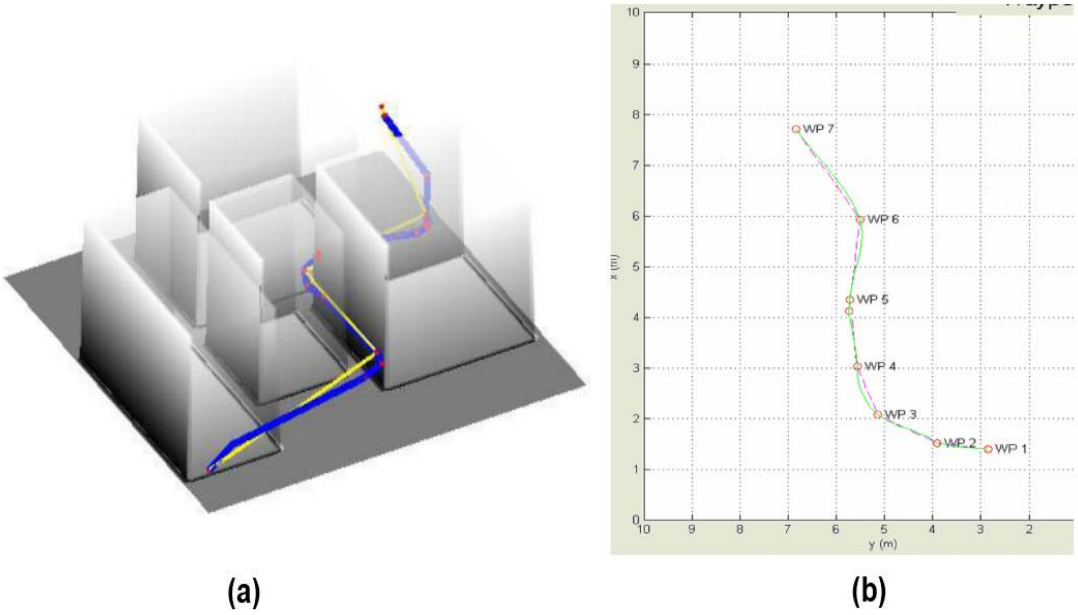


Figura C.9 – Utilização de splines para robôs aéreos. (a) Trajetória de um helicópteros não tripulados para vôos em ambientes 3D densos (KOYUNCU e INALHAN, 2008); (b) Trajetória de um dirigível dado por um conjunto de oito pontos de controle (SHARMA, KULCZYCKI e ELFES, 2007).

No método utilizando cálculos analíticos e geométricos, utilizam-se as equações cinemáticas da plataforma robótica para calcular as velocidades dos atuadores para o robô deslocar-se entre dois pontos através de formas geométricas, um exemplo clássico são as trajetórias calculadas através do caminho de Dubins.

O caminho de Dubins é um método desenvolvido por Dubins (1957) que calcula uma trajetória entre duas posições através de formas geométrica, de modo que para curtos deslocamentos, a trajetória é calculada através de três círculos (CCC), e em longos deslocamentos, através de dois círculos e uma reta (CLC), como mostrado na Figura C.10.

Segundo Alves Neto (2008), o caminho de Dubins é uma técnica utilizada em diversos trabalhos devido, principalmente, ao resultado final e ao mínimo gasto de energia. Porém, este método necessita de uma grande complexidade computacional e produz ocasionalmente uma trajetória com mudanças bruscas, podendo apresentar certas descontinuidades.

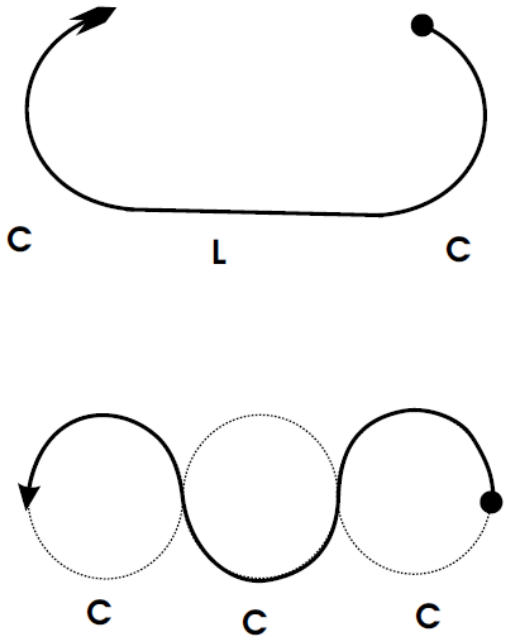


Figura C.10 – Tipos de trajetória alcançadas com o caminho de Dubins (SHANMUGAVEL, 2007).

Portanto, a utilização do caminho de Dubins pode ser considerada muito eficaz, como pode ser visto na Figura C.11, produzindo uma trajetória de menor caminho, contudo, as limitações desta técnica devem ser consideradas, possivelmente no que diz respeito à especificação dos pontos escolhidos.

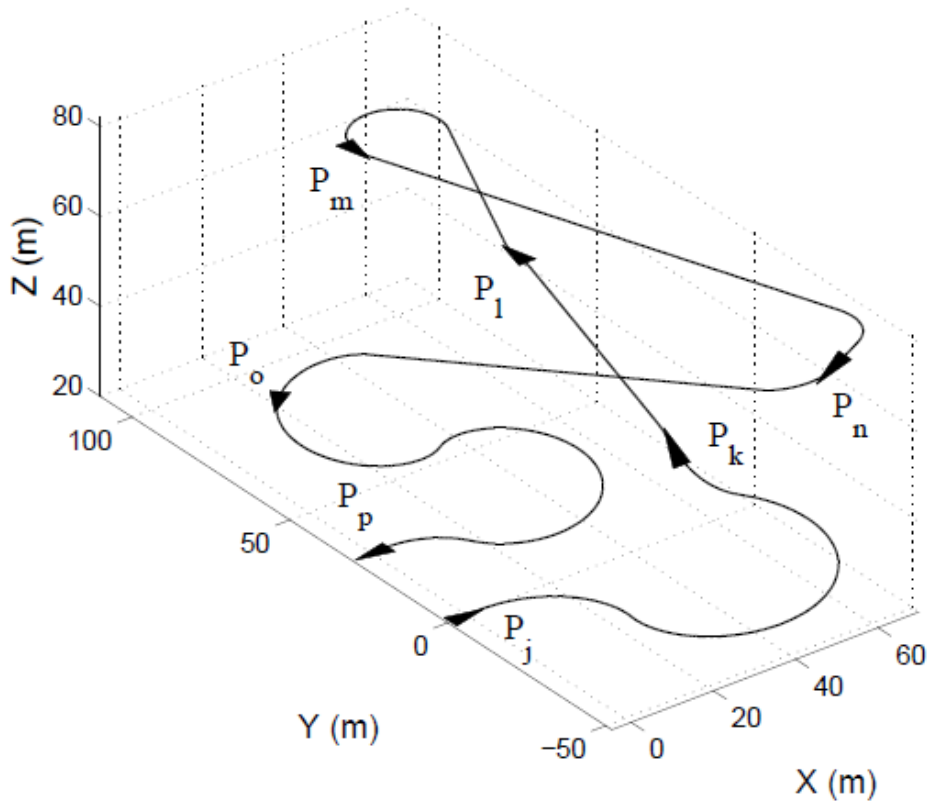


Figura C.11 – Trajetória obtida através do caminho de Dubins (ALVES NETO, 2008).

C.3 Conclusão

O planejamento de movimento é composto por um planejamento de caminhos e um planejamento de trajetórias. O planejamento de caminhos geralmente são constituído por algoritmos de busca, como o A*, D*, BFS e Algoritmo de Dijkstra. O planejamento de trajetória é o módulo responsável em converter as coordenadas pelo qual o robô deve passar em velocidades, acelerações e direções, para isso pode-se destacar duas técnicas: β-splines e o caminho de Dubins.

Através do estudo realizado, conclui-se que todos os algoritmos de buscas apresentados fornecem o resultado desejado no planejamento de caminho de maneira ótima, mas o que os diferencia entre si é o tempo de processamento e sua utilização, assim, neste trabalho, optou-se pela utilização do algoritmo de Dijkstra, devido ao baixo tempo de processamento e a utilização de grafos.

Em relação ao planejamento de trajetória, notou-se que o uso de β-splines fornece uma alta precisão, porém, é necessária uma localização precisa, um alto custo computacional, e a possível hibridização do sistema torna-se complicada, pois como é necessária uma constante mudança de velocidade, o sistema necessitaria conferir a parte deliberativa com recorrência, tornando o sistema lento. A utilização do caminho de Dubins mostrou-se interessante, pois apesar de possuir uma precisão menor e um alto custo computacional elevado, seu custo computacional é menor se comparado a utilização de β-splines e, além disso, facilita a hibridização do sistema, já que a divisão do caminho em três trechos implica em poucas mudanças de velocidade. Entretanto, o caminho determinado pelo caminho de Dubins, por ser sempre o menor caminho, normalmente não segue a trajetória desejada, dificultando a sua utilização numa pista. Portanto, considerando essas características, optou-se pela implementação de um planejamento de trajetória baseado no caminho de Dubins que, ao invés de utilizar três trechos de curvas, utiliza apenas uma curva para alcançar cada sub-objetivo desejado.

ANEXO D – Navegação

A navegação é uma das competências mais desafiadoras exigidas de um robô móvel, de modo que seu sucesso exige sucesso de quatro blocos distintos: a percepção, a localização, a cognição e o controle de movimento; onde na percepção, o robô deve interpretar as leituras obtidas pelos seus sensores para extrair dados significativos; na localização, o robô deve determinar a sua posição no ambiente; na cognição, decidindo como agir para atingir seus objetivos; e no controle de movimento, o robô deve modular as saídas dos motores para alcançar a trajetória desejada (SIEGWART e NOURBAKHSH, 2004).

Murphy (2000) concorda, e acrescenta que a navegação prossegue sendo uma das funções mais desafiadoras para implementar, pois abrange praticamente tudo sobre inteligência artificial robótica: percepção, ação, planejamento, arquiteturas, hardware, eficiência computacional e resolução de problemas.

O problema da navegação de robôs móveis é amplo e complexo, pois o robô pode se encontrar num ambiente estático com poucos objetos ou num ambiente extremamente dinâmico com diversos objetos em movimento, e, o objetivo do robô pode ser desde atingir um local determinado, seguir uma trajetória pré-determinada, ou ainda, explorar e mapear um local desconhecido (IBRAHIM e FERNANDES, 2004).

A navegação robótica consiste em planejar e executar uma trajetória de maneira a atingir um objetivo, modificando-a conforme necessário para evitar obstáculos inesperados. O planejamento deve produzir uma seqüência de ações com o qual o robô alcance o objetivo de maneira eficiente e segura, e na execução, o robô deve verificar o movimento do robô e monitorar as mudanças no ambiente (CROWLEY, 1984).

O planejamento pode envolver dois métodos diferentes: o planejamento global e o local, podendo optar pela utilização de apenas um ou combinar os dois métodos para a obtenção de um melhor resultado, dependendo do ambiente em que o robô está localizado.

Segundo Levi (1987), o planejamento global define o caminho que guia o robô a cada um dos sub-objetivos determinados pelo controle de missão para atingir o objetivo final, e o planejamento local soluciona as obstruções no caminho planejado pelo robô, para que este possa determinar a trajetória a ser seguida. Diferentemente do planejamento global onde o planejamento do caminho geralmente é efetuado antes que o robô desempenhe a tarefa, o planejamento local ocorre em tempo de execução. Contudo, a navegação num ambiente conhecido torna desnecessário o planejamento local, porém, à medida que o conhecimento do ambiente diminui, a necessidade do planejamento local aumenta (RIBEIRO, 2007).

Murphy (2000) separou a navegação em duas categorias de métodos diferentes: a navegação topológica e a métrica, também conhecidas como navegação qualitativa e quantitativa.

A navegação topológica baseia-se na navegação humana, sendo freqüentemente vista como mais simples e natural para robôs baseados em comportamentos. Por exemplo, quando uma pessoa perdida deseja ir a algum lugar e solicita ajuda a outra pessoa, esta pessoa responde dizendo que a pessoa perdida deveria “ir até a porta”, “virar à esquerda”, “seguir o corredor até a terceira porta à direita” e “entrar”. Portanto, desde que o robô conheça uma porta, um corredor, etc., ele atingirá seu objetivo (MURPHY, 2000).

Na navegação topológica as representações dos caminhos podem ser Relacional, focando nas representações gráficas da memória espacial, algo como a idéia de conectar os pontos; ou Associativa, focada numa percepção atrelada com a localização de forma que se assemelha a conexão da percepção com a ação encontrados em comportamentos reflexivos (MURPHY, 2000).

Da mesma forma que na navegação topológica, a navegação métrica procura encontrar um caminho para atingir seu objetivo, porém, estes métodos se diferenciam na forma em que suas técnicas são utilizadas. A maior diferença entre elas, é que as técnicas métricas produzem um caminho, de acordo com medidas, enquanto as técnicas topológicas

produzem os caminhos através de locais identificáveis. Assim, os sub-objetivos das técnicas métricas são dados através de coordenas (x, y) e das topológicas através de locais, como uma porta, um pilar, uma mesa, etc (MURPHY, 2000).

Os robôs reativos apresentaram comportamentos para deslocarem-se pelo ambiente sem colidir, mas a navegação é uma habilidade mais significativa e requer deliberação, pois em alguns ambientes complexos, a pressuposição de não colisão será insuficiente para garantir o sucesso da tarefa. Assim, um robô necessita ser capaz de planejar como chegar num local desejado (MURPHY, 2000).

Portanto, a navegação de robôs móveis pode ser subdividida em cinco etapas:

- Percepção do ambiente: Através da leitura e processamento dos sinais dos sensores, o robô é capaz de "perceber" o ambiente, atualizando e/ou criando o modelo interno do mundo, e reagir de forma coerente aos eventos ocorridos no ambiente;
- Mapeamento e Localização: Construção e utilização do modelo interno do mundo para localizar-se e determinar a posição e orientação do objetivo;
- Planejamento do caminho: Através do modelo interno do mundo o robô deve ser apto a elaborar uma seqüência de ações que ele deve realizar de maneira a atingir seu objetivo, estas ações são geralmente relacionadas aos locais pelos quais o robô deve percorrer;
- Planejamento de trajetória: Nesta etapa o robô computa quais movimentos os atuadores devem realizar para que o robô atinja cada sub-objetivo;
- Execução da trajetória: Utilizando o planejamento da trajetória e a percepção do ambiente, o robô realiza o controle dos atuadores, podendo modificar os movimentos planejados de acordo com a condição atual do ambiente, sem que seja necessário um novo planejamento.

D.1 Sistemas de Navegação Existentes

Atualmente existem alguns sistemas de navegação idealizados e testados, todos eles possuem algumas ou todas as etapas citadas anteriormente, dependendo da arquitetura

utilizada, porém, diferenciam-se entre si pelas abordagens utilizadas em cada uma das etapas, como por exemplo, o tipo de sensor utilizado na percepção do ambiente.

Ribeiro (2007) elaborou um sistema de navegação para uma cadeira móvel inteligente (CaMIn). A partir de uma planta arquitetônica foram estabelecidos alguns "*checkpoints* energéticos", dessa forma, utilizando o algoritmo de Dijkstra o robô seria capaz de planejar um caminho para atingir seu objetivo. Utilizando sensores ultrasom, campos potenciais artificiais e lógica fuzzy o robô navegaria através dos *checkpoints* planejados até atingir o local desejado. Na Figura D.1 é mostrada uma simulação que demonstra a utilização do algoritmo de Dijkstra na escolha dos "*checkpoints* energéticos" e a empregabilidade dos campos potenciais no desvio de obstáculos.

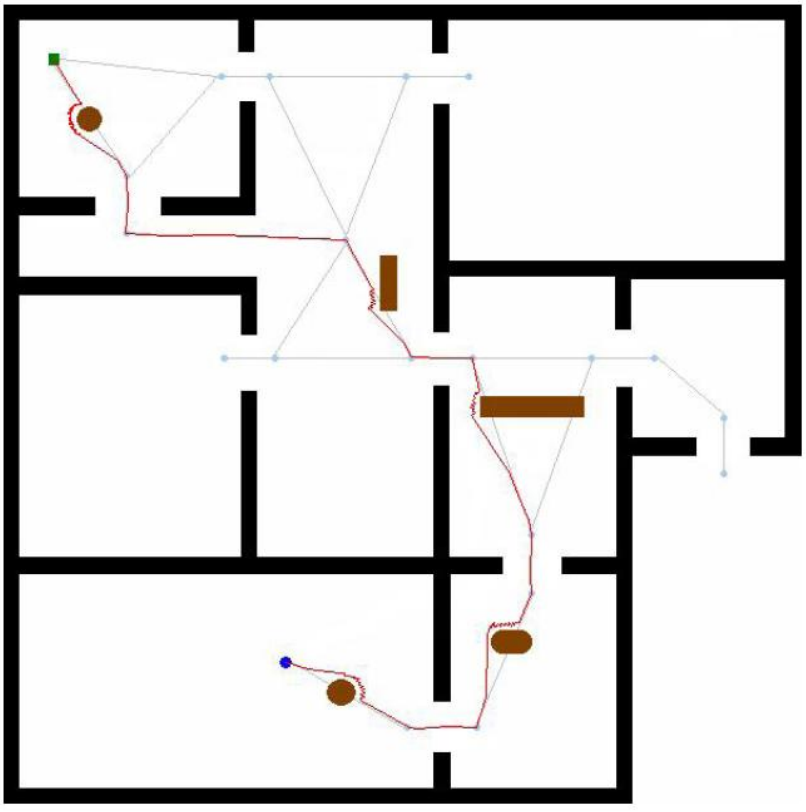


Figura D.1 – Simulação utilizando os "checkpoints energéticos"da CaMIn (Ribeiro, 2007).

Santos (2009) desenvolveu um sistema de navegação baseado na arquitetura híbrida AuRA. O *software Mapper3basic* foi utilizado para decompor uma planta arquitetônica em

grids. O sistema gera o plano a ser realizado através do software ItSIMPLE, e depois utiliza um algoritmo A* para planejar o caminho necessário para a tingir o objetivo. O sistema foi simulado no *software* MobileSim da ActivMedia.

Ottoni e Lages (2003) simularam e testaram num robô um sistema de navegação para ambientes desconhecidos baseado em sonares de ultrasom. O sistema consiste em decompor o ambiente em grids e utilizar o algoritmo de busca em largura para planejar a trajetória. O controlador emprega linearização por realimentação de estado. O sistema foi testado no robô Twil e um dos resultados obtidos é ilustrado abaixo na Figura D.2.

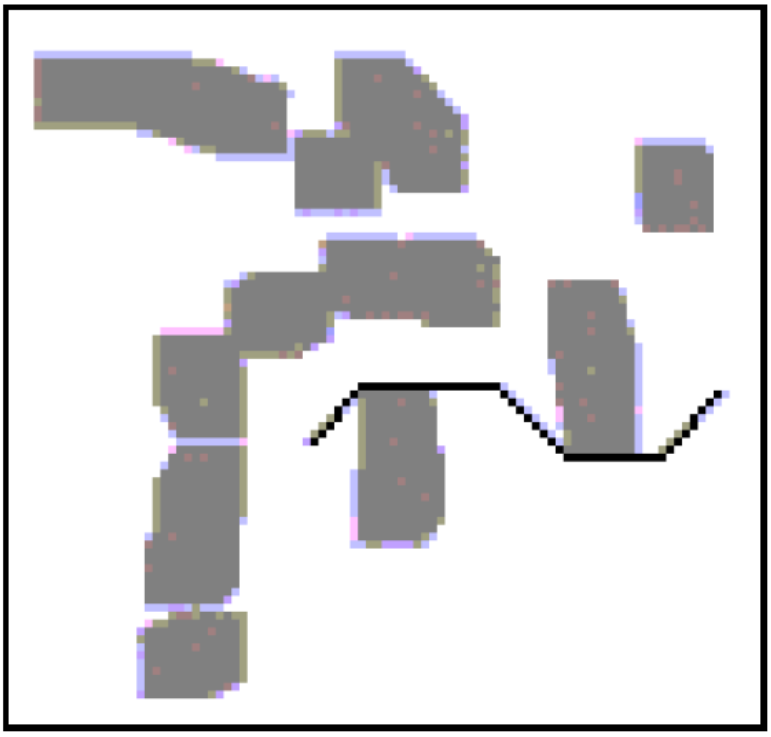


Figura D.2 – Simulação do robô Twil (Ottoni e Lages, 2003).

Pedrosa (2001) desenvolveu um sistema de navegação para a utilização em uma equipe de futebol de robôs. A localização dos robôs é encontrada utilizando uma câmera aérea e o algoritmo de mínimos quadrados. Assim, através do conhecimento da posição atual e da posição desejada calcula-se a trajetória do robô utilizando polinômios paramétricos de terceiro grau. A Figura D.3 mostra um resultado obtido nos experimentos,

onde a linha tracejada indica a trajetória planejada e a linha contínua indica a trajetória realizada pelo robô.

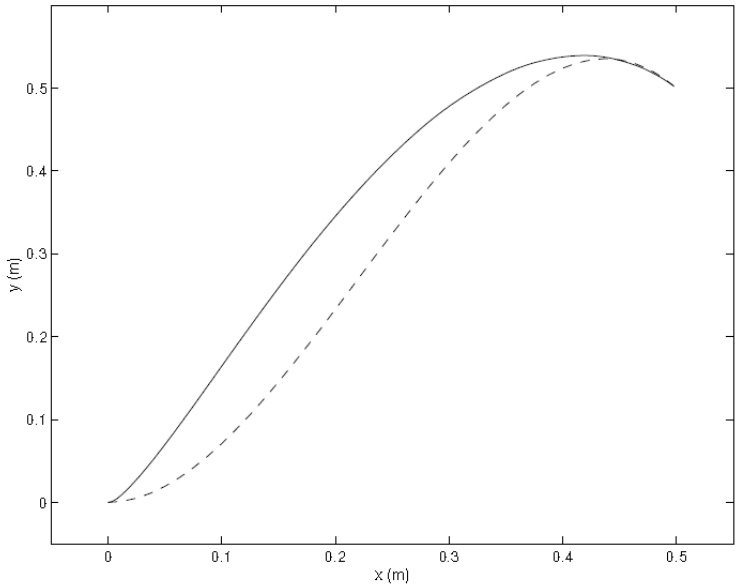


Figura D.3 – Simulação realizada por Pedrosa (2001).

D.2 Conclusão

A navegação robótica é a união de diversas abordagens, cada uma para uma etapa, buscando alcançar um sistema apto a navegar dentro de um ambiente, se locomovendo e atingindo objetivos. Atualmente, existem diversos sistemas de navegação, porém, nenhum deles obteve robustez e generalidade suficiente para ser utilizado no mundo real, portanto, ainda é necessária muita pesquisa nesta área.

ANEXO E – Grafos utilizados no trabalho

E.1 Fábrica

Tabela E.1 – Coordenadas dos nós da Fábrica.

N	x	y
A	2,35	-2
B	0	0
C	2,35	2
D	6	3,55
E	8	5
F	10	3,55
G	13,65	2
H	16	0
I	13,65	-2
J	10	-4,2
K	8	-6
L	6	-4,2
M	8	0

Tabela E.2 – Pesos dos trajetos da Fábrica.

E	N_I	N_F	w_{IF}	w_{FI}	Trajetória
AB	A	B	0,92576	2,46868	T1
BC	B	C	0,92576	2,46868	T2
CM	C	M	1,19871	4,19548	T3
MA	M	A	1,19871	4,19548	T4
MF	M	F	0,81492	2,85223	T5
FE	F	E	0,7411	1,97626	T6
ED	E	D	0,7411	1,97626	T7
DM	D	M	0,81492	2,85223	T8
MG	M	G	1,19871	4,19548	T9
GH	G	H	0,92576	2,46868	T10
HI	H	I	0,92576	2,46868	T11
IM	I	M	1,19871	4,19548	T12

MJ	M	J	0,93038	3,25632	T13
JK	J	K	1,2	3,2	T14
KL	K	L	1,2	3,2	T15
LM	L	M	0,93038	3,25632	T16

Tabela E.3 – Coordenadas dos pontos de controle das trajetórias da Fábrica.

Trajetória		Coordenadas											
T1	x	2,350	2,000	1,500	1,000	0,500	0,000						
	y	-2,000	-1,984	-1,895	-1,694	-1,187	0,000						
T2	x	0,000	0,400	1,000	1,500	2,000	2,350						
	y	0,000	1,187	1,694	1,895	1,984	2,000						
T3	x	2,350	3,000	3,500	4,000	4,500	5,000	5,500	6,000	6,500	7,000	7,500	8,000
	y	2,000	1,952	1,860	1,732	1,591	1,391	1,187	0,968	0,737	0,496	0,249	0,000
T4	x	8,000	7,500	7,000	6,500	6,000	5,500	5,000	4,500	4,000	3,500	3,000	2,350
	y	0,000	-0,249	-0,496	-0,737	-0,968	-1,187	-1,391	-1,591	-1,732	-1,860	-1,952	-2,000
T5	x	8,000	8,398	8,784	9,144	9,467	9,732	9,920	10,000				
	y	0,000	0,500	1,000	1,500	2,000	2,500	3,000	3,550				
T6	x	10,000	9,920	9,569	8,000								
	y	3,550	4,000	4,500	5,000								
T7	x	8,000	6,431	6,080	6,000								
	y	5,000	4,500	4,000	3,550								
T8	x	6,000	6,080	6,268	6,533	6,856	7,216	7,602	8,000				
	y	3,550	3,000	2,500	2,000	1,500	1,000	0,500	0,000				
T9	x	8,000	8,500	9,000	9,500	10,000	10,500	11,000	11,500	12,000	12,500	13,000	13,650
	y	0,000	0,249	0,496	0,737	0,969	1,189	1,391	1,572	1,733	1,860	1,952	2,000
T10	x	13,650	14,000	14,500	15,000	15,500	16,000						
	y	2,000	1,984	1,894	1,694	1,308	0,000						
T11	x	16,000	15,500	15,000	14,500	14,000	13,650						
	y	0,000	-1,308	-1,694	-1,894	-1,984	-2,000						
T12	x	13,650	13,000	12,500	12,000	11,500	11,000	10,500	10,000	9,500	9,000	8,500	8,000
	y	-2,000	-1,952	-1,860	-1,733	-1,572	-1,391	-1,189	-0,969	-0,737	-0,496	-0,249	0,000
T13	x	8,000	8,333	8,659	8,967	9,260	9,519	9,732	9,895	9,988	10,000		

	y	0,000	-0,500	-1,000	-1,500	-2,000	-2,500	-3,000	-3,500	-4,000	-4,200		
T14	x	10,000	9,984	9,843	9,465	8,000							
	y	-4,200	-4,500	-5,000	-5,500	-6,000							
T15	x	8,000	6,535	6,157	6,016	6,000							
	y	-6,000	-5,500	-5,000	-4,500	-4,200							
T16	x	6,000	6,012	6,105	6,268	6,481	6,740	7,033	7,341	7,667	8,000		
	y	-4,200	-4,000	-3,500	-3,000	-2,500	-2,000	-1,500	-1,000	-0,500	0,000		

Tabela E.4 – Pontos de auxílio para as trajetórias da Fábrica

Pontos de auxílio	x	y
$PAUX_{CM}(2)$	7,9956	0,00902
$PAUX_{MA}(1)$	7,9956	-0,0089
$PAUX_{DM}(2)$	7,994	0,00803
$PAUX_{MF}(1)$	8,006	0,00795
$PAUX_{LM}(2)$	7,9918	-0,0058
$PAUX_{MJ}(1)$	8,0081	-0,0058
$PAUX_{IM}(2)$	8,046	-0,0089
$PAUX_{MG}(2)$	8,0043	0,009

E.2 Escritório

Tabela E.5 – Coordenadas dos nós do Escritório.

N	x	y
A	0,191	0,4122
B	0,191	-0,4122
C	1	0
D	2	0
E	3	0
F	4	0
G	5	0

H	6	0
I	2,573	1,382
J	5,286	1,736
K	9	1,5
L	1	2
M	2	2
N	3	2
O	4	2
P	5	2
Q	6	2
R	1	4
S	2	4
T	3	4
U	4	4
V	5	4
W	6	4
X	0,1183	-5,809
Y	1	6
Z	2	6
A1	3	6
A2	4	6
A3	5	6
A4	6	6
A5	9	1,5
D'	2	-0,6
E'	3	-0,6
F'	4	-0,6
G'	5	-0,6
M'	2	-2,6
N'	3	-2,6
O'	4	-2,6
P'	5	-2,6
S'	2	-4,6

T'	3	-4,6
U'	4	-4,6
V'	5	-4,6
Z'	2	-6,6
A1'	3	-6,6
A2'	4	-6,6
A3'	5	-6,6

Tabela E.6 – Pesos dos trajetos do Escritório.

E	N_I	N_F	w_{IF}	w_{FI}	Trajetória
CA	C	A	0,27239	0,72637	T1
CB	C	B	0,27239	0,72637	T2
CL	C	L	0	1	T3
DC	D	C	0	0,5	T4
ED	E	D	0	0,5	T5
FE	F	E	0	0,5	T6
FI	F	I	0	0,99326	T7
GF	G	F	0	0,5	T8
GJ	G	J	0	0,8797	T9
HG	H	G	0	0,5	T10
KH	K	H	0	1,67705	T11
HQ	H	Q	0	1	T12
QP	Q	P	0	0,5	T13
PO	P	O	0	0,5	T14
ON	O	N	0	0,5	T15
NM	N	M	0	0,5	T16
ML	M	L	0	0,5	T17
LR	L	R	0	1	T18
QA5	Q	A5	0	1,80278	T19
QW	Q	W	0	1	T20
WV	W	V	0	0,5	T21
VU	V	U	0	0,5	T22
UT	U	T	0	0,5	T23
TS	T	S	0	0,5	T24
SR	S	R	0	0,5	T25

RY	R	Y	0	1	T26
WA4	W	A4	0	1	T27
A4A5	A4	A5	0	1,80278	T28
A4A3	A4	A3	0	0,5	T29
A3A2	A3	A2	0	0,5	T30
A2A1	A2	A1	0	0,5	T31
A1Z	A1	Z	0	0,5	T32
ZY	Z	Y	0	0,5	T33
YX	Y	X	0	0,45108	T34
DD'	D	D'	0	0,3	T35
EE'	E	E'	0	0,3	T36
FF'	F	F'	0	0,3	T37
GG'	G	G'	0	0,3	T38
MM'	M	M'	0	0,3	T39
NN'	N	N'	0	0,3	T40
OO'	O	O'	0	0,3	T41
PP'	P	P'	0	0,3	T42
SS'	S	S'	0	0,3	T43
TT'	T	T'	0	0,3	T44
UU'	U	U'	0	0,3	T45
VV'	V	V'	0	0,3	T46
ZZ'	Z	Z'	0	0,3	T47
A1A1'	A1	A1'	0	0,3	T48
A2A2'	A2	A2'	0	0,3	T49
A3A3'	A3	A3'	0	0,3	T50

Tabela E.7 – Coordenadas dos pontos de controle das trajetórias do Escritório.

Trajetória		Coordenadas				
T1	x	1,0000	0,7989	0,5991	0,0401	0,1910
	y	0,0000	0,0204	0,0859	0,1994	0,4187
T2	x	1,0000	0,7989	0,5991	0,0401	0,1910
	y	0,0000	-0,0204	-0,0859	-0,1994	-0,4187
T3	x	1,0000	1,0000	1,0000	1,0000	1,0000
	y	0,0000	-0,5000	-1,0000	-1,5000	-2,0000
T4	x	2,0000	1,7500	1,5000	1,2500	1,0000
	y	0,0000	0,0000	0,0000	0,0000	0,0000
T5	x	3,0000	2,7500	2,5000	2,2500	2,0000

	y	0,0000	0,0000	0,0000	0,0000	0,0000
T6	x	4,0000	3,7500	3,5000	3,2500	3,0000
	y	0,0000	0,0000	0,0000	0,0000	0,0000
T7	x	4,0000	3,7960	3,3930	3,0030	2,5730
	y	0,0000	0,0186	0,1711	0,5052	1,3820
T8	x	5,0000	4,7500	4,5000	4,2500	4,0000
	y	0,0000	0,0000	0,0000	0,0000	0,0000
T9	x	5,0000	5,0500	5,1010	5,2050	5,2860
	y	0,0000	0,7664	1,0800	1,5120	1,7630
T10	x	6,0000	5,7500	5,5000	5,2500	5,0000
	y	0,0000	0,0000	0,0000	0,0000	0,0000
T11	x	9,0000	8,5000	7,7430	7,0000	6,0000
	y	1,5000	0,6770	0,2792	0,0859	0,0000
T12	x	6,0000	6,0000	6,0000	6,0000	6,0000
	y	0,0000	-0,5000	-1,0000	-1,5000	-2,0000
T13	x	6,0000	5,7500	5,5000	5,2500	5,0000
	y	-2,0000	-2,0000	-2,0000	-2,0000	-2,0000
T14	x	5,0000	4,7500	4,5000	4,2500	4,0000
	y	-2,0000	-2,0000	-2,0000	-2,0000	-2,0000
T15	x	4,0000	3,7500	3,5000	3,2500	3,0000
	y	-2,0000	-2,0000	-2,0000	-2,0000	-2,0000
T16	x	3,0000	2,7500	2,5000	2,2500	2,0000
	y	-2,0000	-2,0000	-2,0000	-2,0000	-2,0000
T17	x	2,0000	1,7500	1,5000	1,2500	1,0000
	y	-2,0000	-2,0000	-2,0000	-2,0000	-2,0000
T18	x	1,0000	1,0000	1,0000	1,0000	1,0000
	y	-2,0000	-2,5000	-3,0000	-3,5000	-4,0000
T19	x	6,0000	7,0000	7,7430	8,5000	9,0000
	y	-2,0000	-2,1140	-2,3720	-2,9000	-4,0000
T20	x	6,0000	6,0000	6,0000	6,0000	6,0000
	y	-2,0000	-2,5000	-3,0000	-3,5000	-4,0000
T21	x	6,0000	5,7500	5,5000	5,2500	5,0000
	y	-4,0000	-4,0000	-4,0000	-4,0000	-4,0000
T22	x	5,0000	4,7500	4,5000	4,2500	4,0000
	y	-4,0000	-4,0000	-4,0000	-4,0000	-4,0000
T23	x	4,0000	3,7500	3,5000	3,2500	3,0000
	y	-4,0000	-4,0000	-4,0000	-4,0000	-4,0000

T24	x	3,0000	2,7500	2,5000	2,2500	2,0000
	y	-4,0000	-4,0000	-4,0000	-4,0000	-4,0000
T25	x	2,0000	1,7500	1,5000	1,2500	1,0000
	y	-4,0000	-4,0000	-4,0000	-4,0000	-4,0000
T26	x	1,0000	1,0000	1,0000	1,0000	1,0000
	y	-4,0000	-4,5000	-5,0000	-5,5000	-6,0000
T27	x	6,0000	6,0000	6,0000	6,0000	6,0000
	y	-4,0000	-4,5000	-5,0000	-5,5000	-6,0000
T28	x	6,0000	7,0000	7,7430	8,5000	9,0000
	y	-6,0000	-5,8860	-5,6280	-5,1000	-4,0000
T29	x	6,0000	5,7500	5,5000	5,2500	5,0000
	y	-6,0000	-6,0000	-6,0000	-6,0000	-6,0000
T30	x	5,0000	4,7500	4,5000	4,2500	4,0000
	y	-6,0000	-6,0000	-6,0000	-6,0000	-6,0000
T31	x	4,0000	3,7500	3,5000	3,2500	3,0000
	y	-6,0000	-6,0000	-6,0000	-6,0000	-6,0000
T32	x	3,0000	2,7500	2,5000	2,2500	2,0000
	y	-6,0000	-6,0000	-6,0000	-6,0000	-6,0000
T33	x	2,0000	1,7500	1,5000	1,2500	1,0000
	y	-6,0000	-6,0000	-6,0000	-6,0000	-6,0000
T34	x	1,0000	0,8080	0,6023	0,4044	0,1183
	y	-6,0000	-5,9920	-5,9640	-5,9180	-5,8090
T35	x	2,0000	2,0000			
	y	0,0000	-0,6000			
T36	x	3,0000	3,0000			
	y	0,0000	-0,6000			
T37	x	4,0000	4,0000			
	y	0,0000	-0,6000			
T38	x	5,0000	5,0000			
	y	0,0000	-0,6000			
T39	x	2,0000	2,0000			
	y	-2,0000	-2,6000			
T40	x	3,0000	3,0000			
	y	-2,0000	-2,6000			
T41	x	4,0000	4,0000			
	y	-2,0000	-2,6000			
T42	x	5,0000	5,0000			

	y	-2,0000	-2,6000			
T43	x	2,0000	2,0000			
	y	-4,0000	-4,6000			
T44	x	3,0000	3,0000			
	y	-4,0000	-4,6000			
T45	x	4,0000	4,0000			
	y	-4,0000	-4,6000			
T46	x	5,0000	5,0000			
	y	-4,0000	-4,6000			
T47	x	2,0000	2,0000			
	y	-6,0000	-6,6000			
T48	x	3,0000	3,0000			
	y	-6,0000	-6,6000			
T49	x	4,0000	4,0000			
	y	-6,0000	-6,6000			
T50	x	5,0000	5,0000			
	y	-6,0000	-6,6000			

Tabela E.8 – Pontos de auxílio para as trajetórias do Escritório.

Pontos de auxílio	x	y
$PAUX_{CA}(1)$	0,99	0
$PAUX_{CB}(1)$	0,99	0
$PAUX_{YX}(1)$	0,99	-6
$PAUX_{FI}(1)$	3,99	4,40E-05
$PAUX_{GK}(1)$	5	0,01
$PAUX_{KH}(2)$	6,01	0
$PAUX_{QA5}(1)$	6,01	-2
$PAUX_{A4A5}(2)$	6,01	-6
$PAUX_{DC}(1)$	1,99	0
$PAUX_{ED}(2)$	2,01	0
$PAUX_{DD'}(1)$	2	-0,01
$PAUX_{ED}(1)$	2,99	0
$PAUX_{FE}(2)$	3,01	0

$PAUX_{EE'}(1)$	3	-0,01
$PAUX_{FE}(1)$	3,99	0
$PAUX_{GF}(2)$	4,01	0
$PAUX_{FF'}(1)$	4	-0,01
$PAUX_{GF}(1)$	4,99	0
$PAUX_{IG}(2)$	5,01	0
$PAUX_{GG'}(1)$	5	-0,01
$PAUX_{ML}(1)$	1,99	-2
$PAUX_{NM}(2)$	2,01	-2
$PAUX_{MM'}(1)$	2	-2,01
$PAUX_{NM}(1)$	2,99	-2
$PAUX_{ON}(2)$	3,01	-2
$PAUX_{NN'}(1)$	3	-2,01
$PAUX_{ON}(1)$	3,99	-2
$PAUX_{PO}(2)$	4,01	-2
$PAUX_{OO'}(1)$	4	-2,01
$PAUX_{PO}(1)$	4,99	-2
$PAUX_{QP}(2)$	5,01	-2
$PAUX_{PP'}(1)$	5	-2,01
$PAUX_{SR}(1)$	1,99	-4
$PAUX_{TS}(2)$	2,01	-4
$PAUX_{SS'}(1)$	2	-4,01
$PAUX_{TS}(1)$	2,99	-4
$PAUX_{UT}(2)$	3,01	-4
$PAUX_{TT'}(1)$	3	-4,01
$PAUX_{UT}(1)$	3,99	-4
$PAUX_{VU}(2)$	4,01	-4
$PAUX_{UU'}(1)$	4	-4,01
$PAUX_{VU}(1)$	4,99	-4
$PAUX_{WV}(2)$	5,01	-4
$PAUX_{VV'}(1)$	5	-4,01
$PAUX_{ZY}(1)$	1,99	-6

$PAUX_{A1Z}(2)$	2,01	-6
$PAUX_{ZZ'}(1)$	2	-6,01
$PAUX_{A1Z}(1)$	2,99	-6
$PAUX_{A2A1}(2)$	3,01	-6
$PAUX_{A1A1'}(1)$	3	-6,01
$PAUX_{A2A1}(1)$	3,99	-6
$PAUX_{A3A2}(2)$	4,01	-6
$PAUX_{A2A2'}(1)$	4	-6,01
$PAUX_{A3A2}(1)$	4,99	-6
$PAUX_{A4A3}(2)$	5,01	-6
$PAUX_{A3A3'}(1)$	5	-6,01

E.3 Sistema de Tubulação

Tabela E.9 – Coordenadas dos nós do Sistema de Tubulação.

N	x	y
A	-2	6
B	-2	5
C	3	7
D	3	6
E	3	5
F	3	4
G	6	5
H	6	4
I	10	5
J	10	4
K	-2	2
L	-2	1
M	0	2
N	1	2
O	0	1
P	1	1
Q	0	0
R	1	0

S	7	1
T	10	0
U	10	1

Tabela E.10 – Pesos dos trajetos do Sistema de Tubulação.

E	N_I	N_F	w_{IF}	w_{FI}	Trajetória
AD	A	D	0,5	3	T1
BE	B	E	0,5	3	T2
DC	D	C	0,1	0,6	T3
ED	E	D	0,1	0,6	T4
FE	F	E	0,1	0,6	T5
GD	G	D	0,94868	2,52982	T6
HG	H	G	0,1	0,6	T7
FH	F	H	0,3	1,8	T8
EG	E	G	0,3	1,8	T8
GI	G	I	0,4	2,4	T9
HJ	H	J	0,4	2,4	T10
MF	M	F	1,08167	2,88444	T11
NF	N	F	0,84853	2,26274	T12
MN	M	N	0,1	0,6	T13
KM	K	M	0,2	1,2	T14
LO	L	O	0,2	1,2	T15
OM	O	M	0,1	0,6	T16
OP	O	P	0,1	0,6	T17
PN	P	N	0,1	0,6	T18
QO	Q	O	0,1	0,6	T19
RP	R	P	0,1	0,6	T20
HS	H	S	0,63246	2,21359	T21
PS	P	S	1,8	4,8	T22
ST	S	T	0,31623	1,89737	T23
SU	S	U	0,3	1,8	T24

Tabela E.11 – Coordenadas dos pontos de controle das trajetórias do Sistema de Tubulação.

Trajetória		Coordenadas					
T1	x	-2,0000	-1,0000	0,0000	1,0000	2,0000	3,0000
	y	6,0000	6,0000	6,0000	6,0000	6,0000	6,0000

T2	x	-2,0000	-1,0000	0,0000	1,0000	2,0000	3,0000
	y	5,0000	5,0000	5,0000	5,0000	5,0000	5,0000
T3	x	3,0000	3,0000	3,0000			
	y	6,0000	6,5000	7,0000			
T4	x	3,0000	3,0000	3,0000			
	y	5,0000	5,5000	6,0000			
T5	x	3,0000	3,0000	3,0000			
	y	4,0000	4,5000	5,0000			
T6	x	6,0000	5,5000	5,1340	5,0000	4,0000	3,0000
	y	5,0000	5,1340	5,5000	6,0000	6,0000	6,0000
T7	x	6,0000	6,0000	6,0000			
	y	4,0000	4,5000	5,0000			
T8	x	3,0000	4,0000	5,0000	6,0000		
	y	4,0000	4,0000	4,0000	4,0000		
T9	x	3,0000	4,0000	5,0000	6,0000		
	y	5,0000	5,0000	5,0000	5,0000		
T10	x	6,0000	7,0000	8,0000	9,0000	10,0000	
	y	5,0000	5,0000	5,0000	5,0000	5,0000	
T11	x	6,0000	7,0000	8,0000	9,0000	10,0000	
	y	4,0000	4,0000	4,0000	4,0000	4,0000	
T12	x	0,0000	0,0000	0,0000	0,5000	0,8660	1,0000
	y	2,0000	2,5000	3,0000	3,1340	3,5000	4,0000
T13	x	1,0000	1,5000	2,0000	2,5000	2,8660	3,0000
	y	2,0000	2,0000	2,0000	2,1340	2,5000	3,0000
T14	x	0,0000	0,5000	1,0000			
	y	2,0000	2,0000	2,0000			
T15	x	-2,0000	-1,0000	0,0000			
	y	2,0000	2,0000	2,0000			
T16	x	-2,0000	-1,0000	0,0000			
	y	1,0000	1,0000	1,0000			
T17	x	0,0000	0,0000	0,0000			
	y	1,0000	1,5000	2,0000			
T18	x	0,0000	0,5000	1,0000			
	y	1,0000	1,0000	1,0000			
T19	x	1,0000	1,0000	1,0000			
	y	1,0000	1,5000	2,0000			

T20	x	0,0000	0,0000	0,0000			
	y	0,0000	0,5000	1,0000			
T21	x	1,0000	1,0000	1,0000			
	y	0,0000	0,5000	1,0000			
T22	x	6,0000	6,3300	6,6600	7,0000		
	y	4,0000	3,0000	2,0000	1,0000		
T23	x	1,0000	2,0000	3,0000	4,0000	5,0000	6,0000
	y	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000
T24	x	7,0000	7,8660	7,5000	8,0000	9,0000	10,0000
	y	1,0000	0,5000	0,1340	0,0000	0,0000	0,0000
T25	x	7,0000	8,0000	9,0000	10,0000		
	y	1,0000	1,0000	1,0000	1,0000		

Tabela E.12 – Pontos de auxílio para as trajetórias do Sistema de Tubulação.

Pontos de auxílio	x	y
$PAUX_{DA}(1)$	2,99	6
$PAUX_{DC}(1)$	3	6,01
$PAUX_{GD}(2)$	3,01	6
$PAUX_{ED}(2)$	3	5,99
$PAUX_{EB}(1)$	2,99	5
$PAUX_{ED}(1)$	3	5,01
$PAUX_{EG}(2)$	3,01	5
$PAUX_{FE}(2)$	3	4,99
$PAUX_{MF}(1)$	2,99	4
$PAUX_{FE}(1)$	3	4,01
$PAUX_{FH}(2)$	3,01	4
$PAUX_{NF}(2)$	3	3,99
$PAUX_{MN}(2)$	0,99	2
$PAUX_{NF}(1)$	1,01	2
$PAUX_{PN}(2)$	1	1,99
$PAUX_{MF}(1)$	0	2,01
$PAUX_{MN}(1)$	0,01	2

$PAUX_{OM}(2)$	0	1,99
$PAUX_{KM}(2)$	-0,01	2
$PAUX_{OM}(1)$	0	1,01
$PAUX_{OP}(1)$	0,01	1
$PAUX_{QO}(2)$	0	0,99
$PAUX_{LO}(2)$	-0,01	1
$PAUX_{PN}(1)$	1	1,01
$PAUX_{PS}(1)$	1,01	1
$PAUX_{RP}(2)$	1	0,99
$PAUX_{OP}(2)$	0,99	1
$PAUX_{GD}(1)$	6	5,01
$PAUX_{GI}(1)$	6,01	5
$PAUX_{HG}(2)$	6	4,99
$PAUX_{EG}(2)$	5,99	5
$PAUX_{HG}(1)$	6	4,01
$PAUX_{HJ}(1)$	6,01	4
$PAUX_{HS}(1)$	6,0032	3,9905
$PAUX_{FH}(2)$	5,99	4
$PAUX_{HS}(2)$	6,9968	1,0095
$PAUX_{SU}(1)$	7,01	1
$PAUX_{ST}(1)$	7	0,99
$PAUX_{PS}(2)$	6,99	1

ANEXO F – Algoritmos da Simulação

F.1 Planejamento de caminho – Implementação do algoritmo de Dijkstra

```
%Nós do ambiente
%V= [x,y]
%Trajetos entre os nós
%E(i,j)= [Nói Nój peso_ida peso_volta]

s=8;% Nó inicial
f=11;% Nó destino
tamV=length(V);

%Inicialização
for i=1:tamV
    dist(i)=1000+i;
    pai(i)=0;
    if(V(i)==s)
        dist(i)=0;
    end
end

S=[];
Q=V;
distQ=dist;

%Detrminação do melhor caminho
while(~isempty(Q))
    du=min(distQ);
    for i=1:length(V)
        if(dist(i)==du)
            for j=1:length(Q)
                if(Q(j)==i)
                    posu=i;
                    break;
                end
            end

        end
    end

    S=[S;posu];
    Q=V;
    Q(S)=[];
    distQ=dist;
    distQ(S)=[];

    for i=1:length(E)
        if(E(i,1)==posu)
            if(dist((E(i,2)))>(du+E(i,3)))
               dist((E(i,2)))=du+E(i,3);
               pai(E(i,2))=posu;
```

```
            end
        end
        if(E(i,2)==posu)
            if(dist((E(i,1)))>(du+E(i,4)))
               dist((E(i,1)))=du+E(i,4);
               pai(E(i,1))=posu;
            end
        end
    end

end

caminho=[f];
filho=f;
while(filho~=s)
    filho=pai(filho);
    caminho=[caminho;filho];
end

%Determinação dos trajetos utilizados
caminho=caminho(length(caminho):-1:1)

for i=1:(length(caminho)-1)
    for j=1:length(E)
        if((E(j,1)==caminho(i))&&(E(j,2)==caminho(i+1)))
            trajeto(i)=j;
        end
        if((E(j,2)==caminho(i))&&(E(j,1)==caminho(i+1)))
            trajeto(i)=-j;
        end
    end
end
trajeto
```

F.2 Planejamento de trajetória

```
imat=81.56/2/100/0.0225; %correção do fator tempo no matlab
%imat=distância percorrida/tempo/velocidade/raio da roda no teste

%parâmetros do ASURO
r=0.0225;
l=0.1;
%dados de entrada= pontos, ângulo inicial e ângulo final
angini=0/180*pi;
angfim=0/180*pi;
%pontos= pontos de controle
pontos=[0,0;
        5,5];
%os pontos podem ser passados pelo workspace do MATLAB
%pontos=pontos2;
tam=size(pontos);
```

```
pontos(1,3)=angini;
pontos(tam(1),3)=angfim;

%inicializando as variáveis
tind=1;
tempo=0;
beta1=angini;
trajeto=[];
pts(1,:)=pontos(1,:);

%Cálculo das velocidades
for i=2:(tam(1))

    if(pts(1,:)==pontos(i,:))
        marc1=0;
        marc2=0;
    else
        pts(2,:)=pontos(i,:);
        marc1=1;
        marc2=0;
    end


    %se hover um ângulo para posicionamento
    if(pts(2,3)~=0)
        marc2=1;
    end

    while((marc1~=0)||(marc2~=0))


        difx=pts(2,1)-pts(1,1);
        if(difx==0)
            difx=1e-16;
        end
        dify=pts(2,2)-pts(1,2);
        Hdif=sqrt(dify^2+difx^2);
        alpha1=atan2(dify,difx);

        beta2=pi/2-alpha1;
        gama=beta2-beta1;
        dx=Hdif*sin(gama);
        if(dx==0)
            dx=1e-16;
        end
        dy=Hdif*cos(gama);
        sentido=(dy/dx)/abs(dy/dx);
        alpha3=atan(dy/dx);
        alpha2=atan2(dy,dx)+1e-10;
        alpha=alpha2;

        %para mover-se de ré
        %((alpha>=abs((45/180*pi)))&&(alpha<=abs((90/180*pi))))
        %movendo apenas frontalmente
        if((alpha>=(45/180*pi))&&(alpha<=(135/180*pi)))
```

```
            TetR=2*(pi/2-alpha);
            if(TetR==0)
                TetR=1e-16;
            end

            R=dy/tan(TetR)+dx;
            marc1=0;
            pts(1,:)=pts(2,:);
        else
            if(alpha2>=0)
                TetR=pi/2-alpha;
            else
                TetR=-sentido*pi-pi/2-alpha;
            end

            R=0;
        end

        %para posicionamento
        if ((marc2==1)&&(abs(dx)<1e-3)&&(abs(dy)<1e-3))
            ang=pts(2,3);

            if(ang==360)
                ang=0;
            end

            TetR=ang-beta1;
            R=0;
            marc2=0;

        end

         %R=dy/sin(TetR);

         %atualizando posição
         beta1=beta1+TetR;

         if beta1>pi
             beta1=-2*pi+beta1;
         end

         if beta1<-pi
             beta1=2*pi+beta1;
         end



         %calculando distância a ser deslocada pelo centro de massa do
robô
         if(abs(R)>=(l/2))
            distdesl=abs(TetR*(R));
         else
            distdesl=abs(TetR*(R+l/2));
         end
```

```
        %Vn= velocidade média em m/s, podendo ser ajustada como desejada
        %neste caso consideramos o Raio como parâmetro, porém é possível
        % a velocidade constante observando cuidadosamente os pontos de
controle
        Vn=0.3*abs(R)+1e-2;%+0.05;
        %o valor da constante depende apenas do projeto,
        %variando de 0 até a velocidade máima,
        %de acordo com a necessidade da tarefa e do robô

        if(Vn>0.3)
            Vn=0.3;
        end

        tempdesl=distdesl/Vn;
        w=TetR/tempdesl;
        ve=w*(R+l/2);
        vd=w*(R-l/2);

        %velocidade das rodas em rad/s
        we=ve/r;
        wd=vd/r;


        trajeto(tind,:)=[tempo,wd,we,1,0,1,0];
        tind=tind+1;
        tempo=tempo+(tempdesl/imat);
        trajeto(tind,:)=[tempo,wd,we,1,0,1,0];
        tind=tind+1;
    end

end

%O robô fica parado no fim
trajeto(tind,:)=[tempo,0,0,1,0,1,0];
'Parâmetros determinados'
```

F.3 Execução da trajetória

```
%Módulo CDRT
%Determinação da distância entre a posição atual e a pista demarcada
%composta por 5 fases= reta, curva, reta, curva, reta
%
%x,y=posição atual do robô
%erroantes = erro máximo anterior
%faseantes = fase anterior
function [dist,fim,erromaxh,fase] = fcn(x,y,erroantes,faseant)
%distância entre o robô e a pista
%fim = flag de fim de trajetória
%erromaxh= erro máximo atual
```

```
%fase = fase atual

fase=1;
dist1=0;
fim=0;

fase=faseant;

if fase==1
    xfim=0;
    yfim=1.9;

    dist1=x;

    df=sqrt((xfim-x)^2+(yfim-y)^2);
    if abs(df)<1e-3
        fase=2;
    end
    if y>=yfim
        if abs(df)<1e-2
            fase=2;
        end
    end
end
if fase==2
    xfim=0.1;
    yfim=2;
    raio=0.1;
    yref=1.9;

    dist1=raio-sqrt((x-xfim)^2+(y-yref)^2);

    df=sqrt((xfim-x)^2+(yfim-y)^2);
    if abs(df)<1e-3
        fase=3;
    end
    if x>=xfim
        if abs(df)<1e-2
            fase=3;
        end
    end
end
if fase==3
    xfim=2;
    yfim=2;
    dist1=yfim-y;

    df=sqrt((xfim-x)^2+(yfim-y)^2);
    if abs(df)<=1e-3
        fase=4;
        %fim=1;
    end
    if x>=xfim
        if abs(df)<1e-2
            fase=4;
```

```
            %fim=1;
        end
    end

end
if fase==4
    xfim=3;
    yfim=3;
    raio=-1;
    xref=3;

    dist1=raio+sqrt((x-xref)^2+(y-yfim)^2);

    df=sqrt((xfim-x)^2+(yfim-y)^2);
    if abs(df)<1e-3
        fase=5;
    end
    if x>=xfim
        if abs(df)<1e-2
            fase=5;
        end
    end
end

if fase==5
    xfim=3;
    yfim=5;

    dist1=x;

    df=sqrt((xfim-x)^2+(yfim-y)^2);
    if abs(df)<1e-3
        fim=1;
    end
    if y>=yfim
        if abs(df)<1e-2
            fim=1;
        end
    end
end


dist = dist1;


erromaxh=erroantes;
erroh=abs(dist);

if erroh>erroantes
    erromaxh=erroh;
end
```

```
%Módulo Velocidade Reativas
%Determinação das velocidades reativas
%
%dist = dstância determinada pelo Módulo CDRT
%fimant = flag de fim de trajetória anterior
%fim = flaf de fim de trajetória atual
%vmd = velocidade angular calculda deliberativamente para a roda direita
%vme = velocidade angular calculda deliberativamente para a roda esquerda
function [wd,we,fim1]= fcn(dist,fimant,fim,vmd,vme)
%wd = velocidade angular reativa da roda direita
%we = velocidade angular reativa da roda esquerda
%fim1 = flag de saída de fim de trajetória

r=0.0225; % raio da roda
vm=0.3; % velocidade máxima
kr=10; % ganho proporcional


%Cálculo das velocidades
vm=vmd;
vd = vm+kr*dist*vm*2/0.2;
vm=vme;
ve = vm-kr*dist*vm*2/0.2;
fim1=0;


% Ajuste para a velocidade das rodas não ser maior que a velocidade
máxima
if vd>vm
    vd = vm;
end

if ve>vm
    ve = vm;
end
if vd<-vm
    vd = -vm;
end

if ve<-vm
    ve = -vm;
end

if ((fim==1)||(fimant==1))
    fim1=1;
end

%Para o robô no fim da trajetória
if(fim1==1)
    vd=0;
    ve=0;
end
```

```
%Determinação as velocidades reativas das rodas
wd=vd/r;
we=ve/r;
```

ANEXO G – Artigo Publicado

Artigo publicado no Robocontrol 2010, 4º Workshop em Robótica Aplicada e Automação, realizado em Bauru com o título: "***TRAJECTORY PLANNING USING A TOPOLOGICAL MAP FOR DIFFERENTIAL MOBILE ROBOTS*".**

TRAJECTORY PLANNING USING A TOPOLOGICAL MAP FOR DIFFERENTIAL MOBILE ROBOTS

Augusto Seganfredo Mainardi [1], *Alvaro Joffre Uribe Quevedo* [1], *João Maurício Rosário* [1]

[1] Integrated Robotics and Automation Laboratory, Mechanical Engineering, UNICAMP, Campinas S.P., Brazil, (gmainardi, engajuq, rosario)@fem.unicamp.br

Abstract: Mobile robots can be used in various environments such as home, academic and industrial ones, with any application, the trajectory planning becomes an important problem to be addressed by the autonomous vehicle. This paper focuses on the trajectory planning using a topological map for differential mobile robots. For the robot to reach each chosen target first robot's kinematics were calculated and then the geographic and topological map points were defined. The simulation allowed the robot to reach the desired positions proving successful the approach taken.

Keywords: Trajectory planning, topological map, differential drive mobile robot.

1. INTRODUCTION

Development and research in the Mobile robotics field is focused on autonomous and semiautonomous control. The objective is to improve robot navigation while executing tasks in various indoor (offices, factories [1], buildings, houses [2-5], hospitals [6]) or outdoors (space, military, desert and agriculture) environments. The impact in robotics applications continue to grow as they merge and meet with some of our daily needs.

In industrial hazardous scenarios, inspection, maintenance, and transportation require the use of teleoperated or autonomous mobile robots [7], because of their accuracy and freedom mobility within the workspace. A major advantage of mobile devices, is that task execution is not constraint by the target area, various activities can be performed in different situations yielding to robot's scalability and portability offering a wide range of applications.

In [8] the ARK robot (Autonomous Mobile Robot in a Known Environment) navigates through a visual landmark system based on identifiable beacons which were added later to the system. The three wheeled robot is equipped with a touch sensor on its bumper, six sonar and a novel laser based vision sensor (Laser Eye). Tests with the ARK robot were done at the engineering laboratory, where they confirm ARK's capacity for identifying beacons and moving around laboratory.

In order to overcome navigation constraints and docking problems, in [9], a low cost mobile robot platform is developed in which, through the combinations of CCD camera signal measurements, odometer and ultrasonic sensors. To prove the effectiveness of the proposed solution, tests were performed allowing the tracking of free trajectories and successful target docking in the production environment of the company Schoeller-Bregenz.

You et. al [10], developed a home service robot 'ISSAC' for vacuum cleaning and home security, it is composed of twelve ultrasonic, ten and gyroscope sensor. Visual feedback was accomplished through an USB camera integration. The implemented architecture was implemented in an apartment for a week where it cleaned and performed successful surveillance tasks.

This paper proposes a trajectory planning simulation using curves from known topological map points to reduce the execution´s computational cost of the robot. Simulation, of the path planning with curves is going to be used for validating the desired trajectory.

With the industrial environment's blueprint, the topological map can be created from known points defined by the trajectory to be followed over the calculated path. For testing purposes the ASUROTM differential robot is used, so the defined curve, radius and angles, allow solving the

velocities for simulating over the DD&GP toolbox developed for MATLAB®.

This paper is organized as follows: in Section 2 it is given a brief description about the topological map, Section 3 describes the robot analysis. Section 4 explains the trajectory planning proposed, and describes the simulation characteristics. Section 5 presents the simulation perform. Section 6 presents the results, and finally, the conclusions are presented in Section 7.

2. TOPOLOGICAL MAP

A topological map is a summarized description of the configuration of any environment which describes the environment as a collection of places linked by paths. It is used to convey information that is not necessarily focused on precise geography data. The map is simplified to provide a general overview of an area so its information is clearer and easier to interpret and use.

The main advantage of using a topological representation is due to the process of creating maps from experience, the resulted data's consistency defines whether the map was correctly generated or not [11-12].

Contrary to geographic maps where precision and scaling are not relevant parameters when creating topological ones, where the layout of the interest points plays an important role. The core information for implementing a topological map has to be taken from the geographic one, where, depending on the objectives of the mobile robot, some points may be chosen over others. Fig. 1 shows a geographic map over a topological one, in it, circled letters represent the points of interest for moving over a hall and entering into offices.

The topological map is often used to produce maps that convey statistical information about the world, so the user can see roughly where in the space (relatively to world coordinated system) the data is coming from. In some cases, the topological map may be extremely distorted for emphasizing a point or put it within a statistic context.

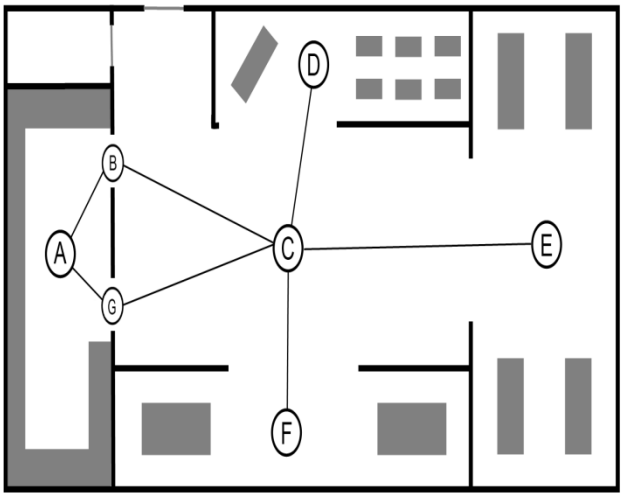


Fig. 1 - Geographic and Topological Map overlay of an environment

3. MOBILE ROBOT ANALYSIS

In order to reach the goal of this paper, the mobile robot is analyzed its simulation model and from its hardware characteristics.

Hardware

The chosen robotics platform for modeling and testing is the ASURO (Another Small Unique Robot from *Oberpfaffenhofen*) (Fig. 2a) [13], designed by the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR) in *Oberpfaffenhofen*, Germany. It is a small, low cost, freely in C programmable mobile robot kit which has been developed for educational.

Based in differential drive, it is composed of one microprocessor, two photodiode-sensor line-follower, six bump sensors, two photodiode-sensors for odometer, two IR sensors for communications and it is driven by two DC motors with gears [14].

Kinematics

The kinematics is calculated and quantified through the geometric relationships that govern the system; its displacement can be calculated to comply with the chosen control strategy and the state space behavior of the system.

The kinematic model of the ASURO is based on its differential drive system shown in Fig. 2b, from which, it can be seen the chosen coordinate system, velocities of wheel right and left, Vr and Vl, respectively, radius of wheel r, angular velocity *w*, distance between wheels *l*, global orientation of the robot θ, curvature's radius *R* and *ICC* (Instantaneous Center of Curvature).

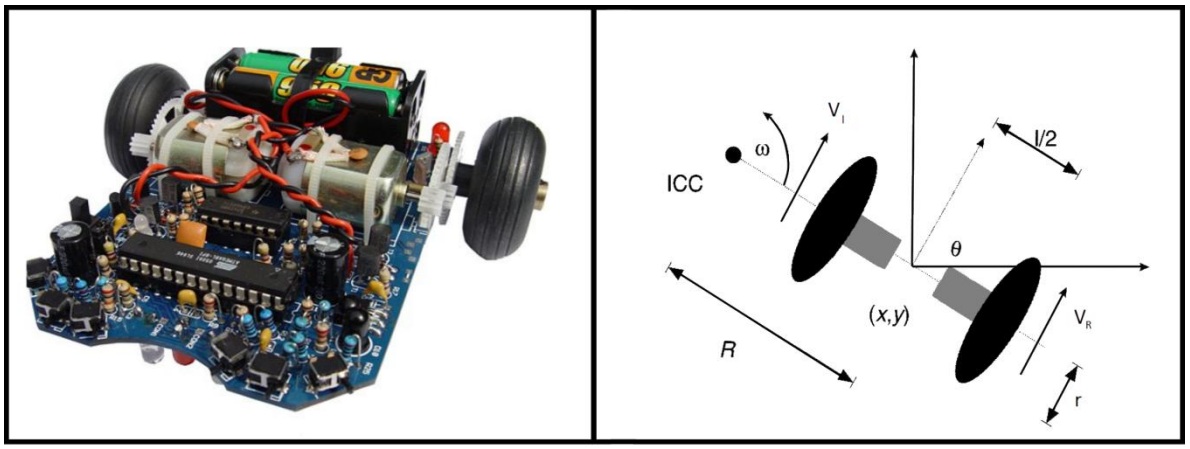


Fig. 2 – (a) Robot Asuro; (b) Differential Drive Kinematics.

For determining the linear velocity around *ICC*, *Vl* and *Vr* are calculated as shown in equation (1) and (2) [15].

$$V_R = \omega * \left(R - \frac{l}{2} \right) \tag{1}$$

$$V_l = \omega * \left(R + \frac{l}{2} \right) \tag{2}$$

Thus, from relating equations (1) and (2), *w* and *R* can be calculated following equations (3) and (4) [15].

$$R = \frac{l}{2} * \left(\frac{V_l + V_R}{V_l - V_R} \right) \tag{3}$$

$$\omega = \frac{V_R - V_l}{l} \tag{4}$$

Therefore, the equation of robot's linear velocity is equation 5 [15]:

$$V = \left(\frac{V_l + V_R}{2} \right) \tag{5}$$

Using the previous calculations (equations (1) to (5)) the kinematic model matrix can be defined as shown in equation (6) [15]:

$$\begin{bmatrix} V_X \\ V_Y \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ \frac{-r}{l} & \frac{r}{l} \end{bmatrix} * \begin{bmatrix} \omega_l \\ \omega_R \end{bmatrix} \tag{6}$$

4. TRAJECTORY PLANNING

The trajectory planning presented in this paper consists in split the path desired in several segments, and calculates the wheel's constant velocities over each segment using equations (1) and (2). For achieving this, *l* is measured, *w* is determined from the desired average velocity, and *R* is calculated trough the two consecutive points A and B (Fig. 3) to follow a curve.

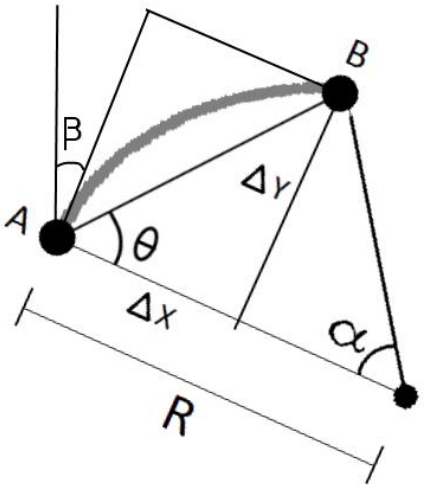


Fig. 3 - Mathematics relations of curvature between A and B

From Fig.3can be seen that α is the curvature's angle, β is the actual orientation, θ is angle's trajectory, ΔX and ΔY are distances in x and y of point B from point A considering ICC as reference, and R is curvature's radius.

So, to calculate *R*, first α is solved (curvature's angle) (Fig. 3); for calculating α an empiric was used approximation (7- 9) and, so, *R* is found trough (10).

$$DIR = \frac{\left(\frac{\Delta Y}{\Delta X}\right)}{abs\left(\frac{\Delta Y}{\Delta X}\right)} \tag{7}$$

$$\theta = \tan\left(\frac{\Delta Y}{\Delta X}\right) \tag{8}$$

$$\alpha = 2 * \left(DIR * \frac{\pi}{2} - \theta\right) \tag{9}$$

$$R = \frac{\Delta Y}{\sin(\alpha)} \tag{10}$$

Having R, the linear velocities are calculated trough (1) and (2), allowing the determination of the angular velocities for each segment of the path as shown in (11) and (12) [15].

$$\omega_R = \frac{V_R}{r} \tag{11}$$

$$\omega_l = \frac{V_l}{r} \tag{12}$$

For solving the i^{th} segment the current orientation has to be updated, so, the angle θ is added to the old orientation, thus, reaching a new β orientation. These angles permit the calculation of the velocities for the new segment.

5. MODEL IMPLEMENTATION AND SIMULATION

For simulating the trajectory, the path points were defined and the velocities calculated. For validation purpose was used the Differential Drive and Global Positioning blockset descript below.

5.1.Simulation Toolbox

The Differential Drive and Global Positioning blockset is a toolbox which can be used with Simulink®, it allows dynamic simulation of differentially driven vehicle robots, this is accomplished through block implemented algorithms related to Global Positioning ones (constraint to dead-reckoning), which are useful in finding out the current position and directions of the vehicle robot [16].

The block set, shown in Fig. 4, has seven different blocks: The PMDC Motor, Gearbox, Motor Driver, Mechanical Dimensions, Positioning, Dual PID, and the Animation blocks. These constitute a complete set of

components that allows modeling a complete differential driven vehicle robot. The obtained model can be tested as a model-driven animation whose parameters can be modified through the GUI. The animation block allows the visualization of the modeled robot, the interaction is done through user modified parameters such as each wheel's angular velocity, and the rotation direction of the motors [16].

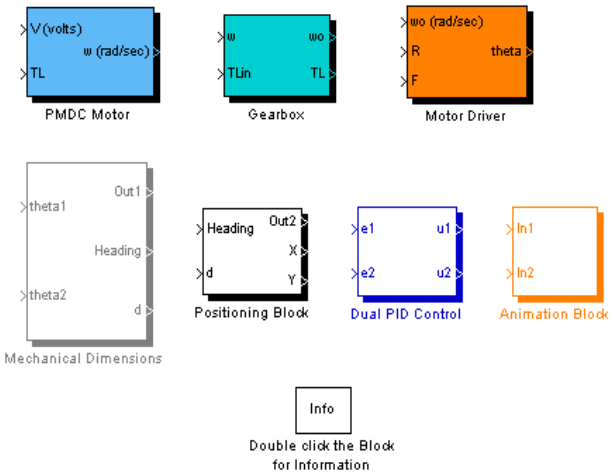


Fig. 4 – Differential Drive and Global Positioning blockset

5.2. Simulation

For implementing the robot model using the toolbox, only the Mechanical Dimensions, Positioning and the Animation blocks are used, the reason for this is that the objective of the simulation is to validate response of calculation of each wheel velocity for the planned trajectory, thus further or additional blocks are not needed. This configuration can be seen in Fig. 5. Mechanical Dimension block was changed in order to meet the parameters of the ASURO for simulating its velocities during the trajectory.

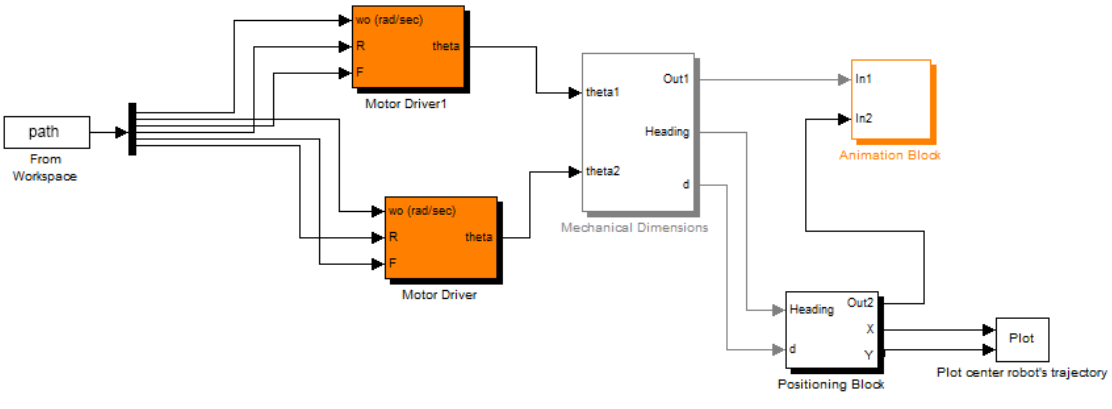


Fig. 5 – Used toolbox configuration in MATLAB®

An industrial scenario is considered for simulation purposes, in its four workbench points A, D, E and F are located as shown in Fig. 1. The ideal path for following the motion between those points is known as a four petals rose or double infinity path, presented in Fig. 6. A simulation parameter for is the rose´s size, so

the adopted configuration allows testing how well the trajectory planner can go through the desired points while varying its position.

Having defined which points are the ones needed from the geographic map, the topological map is created with the required information, four workbenches points at the outer end of each curve, nine points for trajectory planning, additional points for increasing curve accuracy, and additional aid points for obtaining more acute curves. The trajectory points were chosen because they link two different segments of the path. The control points were chosen proportionally among the curved segments, and so, the aid points are set to 0.1, 0.2 or 0.3 in x or y apart from point e for following the path, but, when were used, point C is no longer required.

The trajectory planner validation is tested with a four workbench path, where the target is to go from point A to point F, then to point E, and to point D, for finally returning to A. The second path goes from A to any other, so it goes to point F, then from A to E and finally from A to D.

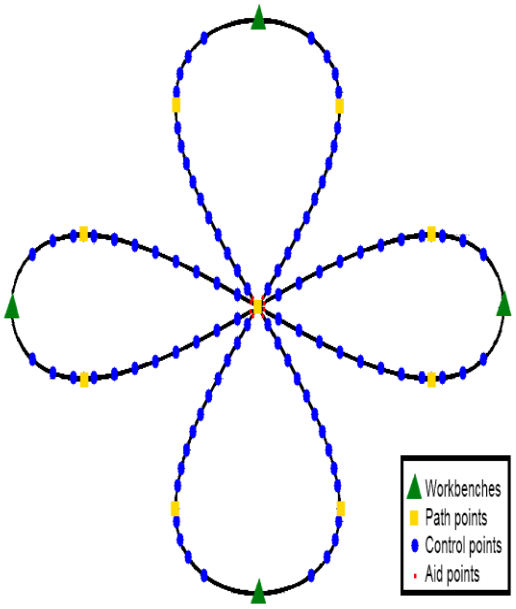


Fig. 6 - Topological map from desired environment

6. RESULTS

First analysis was simulate first path with different number of control points to verify which control points respond better at proposed path. Three simulations were realized, with 17 points (Fig.7), 41 control points (Fig.8) and 124 control points (Fig.9).

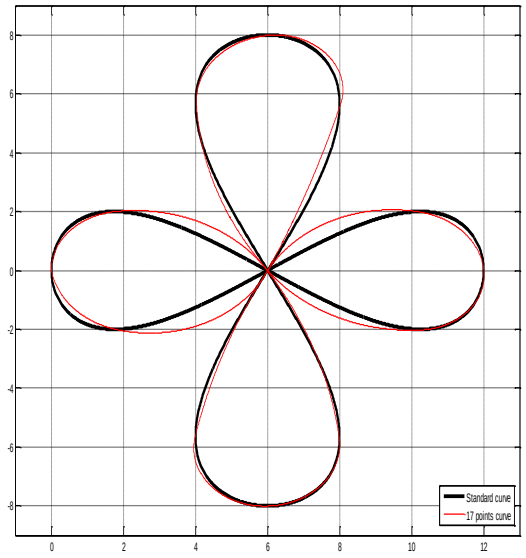


Fig. 7 - First path trajectory with 17 points using point *C*

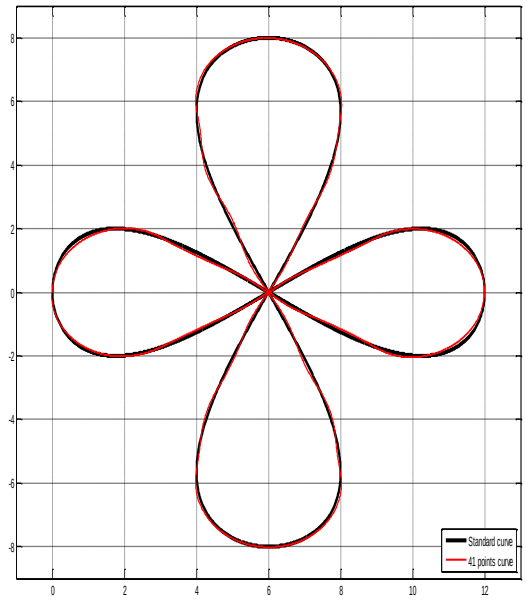


Fig. 8 - First path trajectory with 41 points using aid point 0.1 far from point *C*

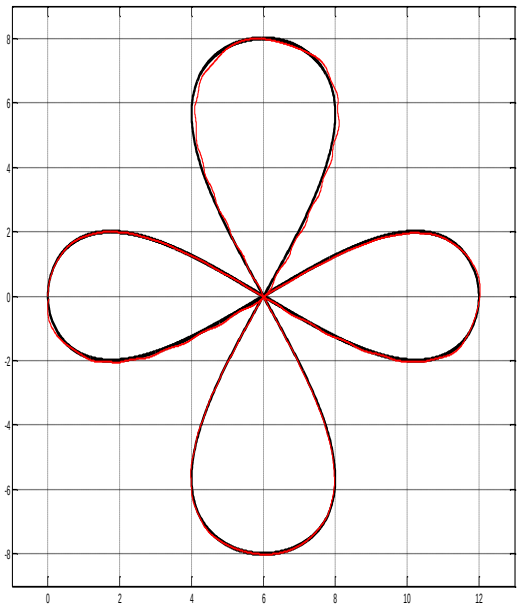


Fig. 9 - First path trajectory with 124 points using aid point 0.1 far from point *C*

The trajectory reached with 17 control points passes through all them, but the robot apart from the target path. Both 41 and 124 points trajectories, followed the path successfully, resulting in a direct relationship of point and accuracy, as the number of them increases so does the precision.

The second analysis uses a different trajectory, the points were chosen with same proportion in the previous trajectory so in order to complete the path, 124 points must be followed. In this case, the trajectories, *A* to *F* (Fig.10), *A* to *E* (Fig.11) and *A* to *D* (Fig.12 and Fig.13). The *A* to *D* trajectory was tested as presented in Fig.12 using the aid points and as seen in Fig.13, going to point E and then rotating.

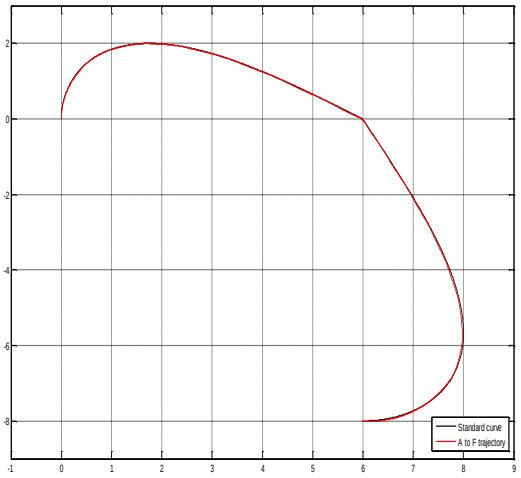


Fig. 10 - Trajectory from point A to F

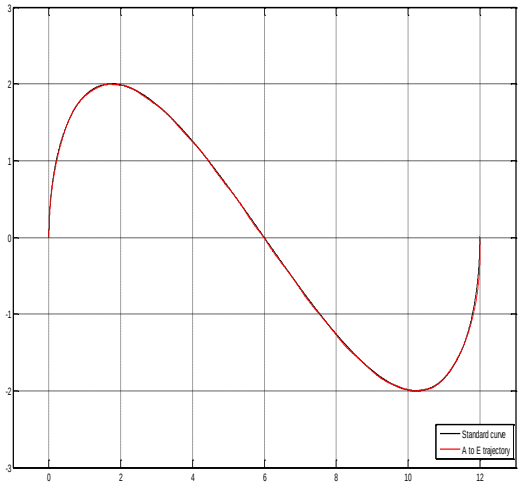


Fig. 11- Trajectory from point A to E

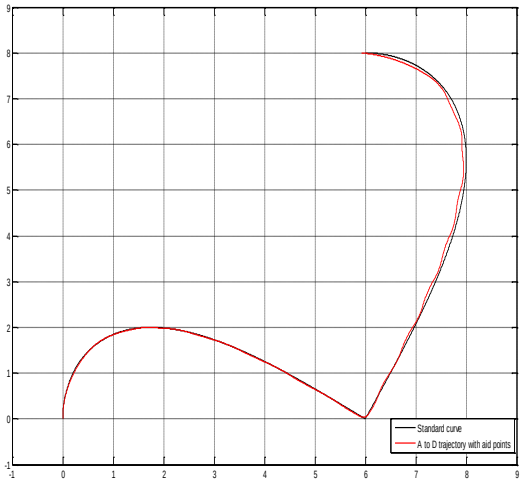


Fig. 12 - Trajectory from point A to D using aid points

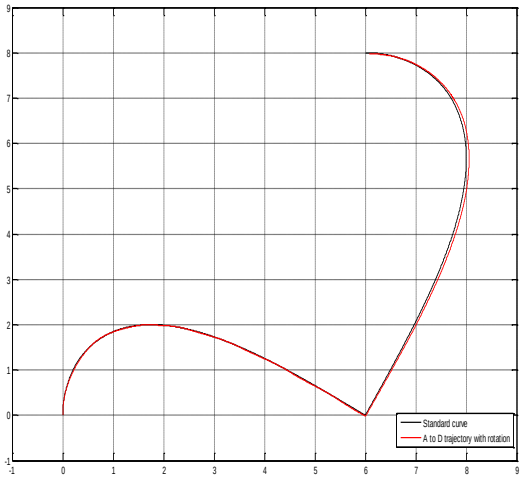


Fig. 3 - Trajectory from A to D with rotation

The path from A to F and from A to E was successfully followed as it was highly accurate with minimal error. Trajectories from A to D were successful too, with the difference, that using the rotation, allowed performing a more accurate trajectory, but it takes more processing and execution time. The time spent while rotating can produce traffic perturbations and conflicts in the production chain.

The tests observed until now used aid points placed at 0.1 in x or y apart from C. Hence, others aid points were tested, 0.2 and 0.3. The rose path was used again with 124 points for trajectory planning and aid points at 0.2 and 0.3, this resulted in Fig. 14 an Fig. 15.

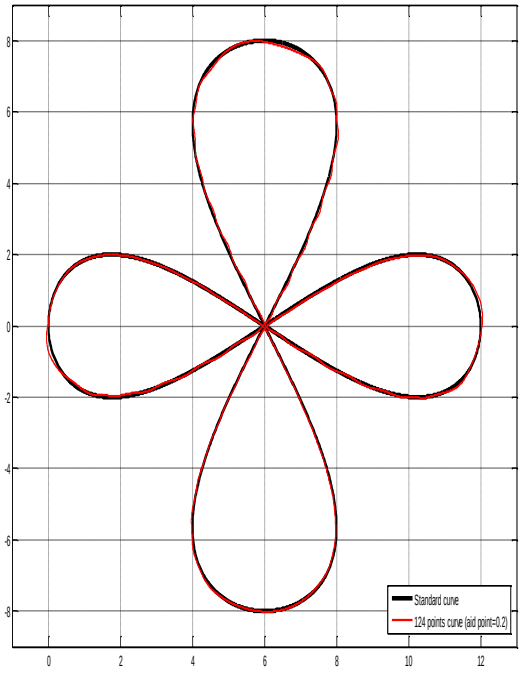


Fig. 4 - First path trajectory with 124 points using aid point 0.2 far from point *C*

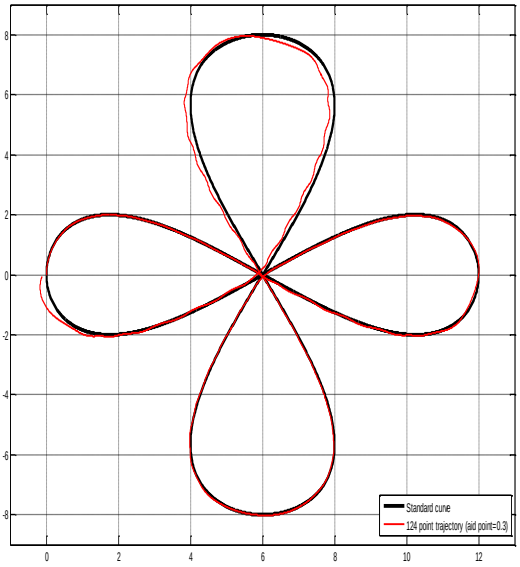


Fig. 5 - First path trajectory with 124 points using aid point 0.3 far from point C

Trajectories reached using 0.2 (Fig.14) and 0.3 (Fig.15) as aid point was satisfactory. However, the trajectory using 0.3 was less accurate, getting out from path and pass far from point C. Meanwhile, the trajectory using 0.2 as aid points reached results more accurate than using 0.1 or 0.3.

For verifying the results, three trajectory simulations from A to F using 0.2 as aid points, each with a different number of points were done. First simulations used 5 points (proportional at 17 points in complete path), second used 11points (proportional at 41 points in complete path) and the third used 31 points (proportional at 124 points in complete path). The results of these simulations are shown in Fig. 16 below.

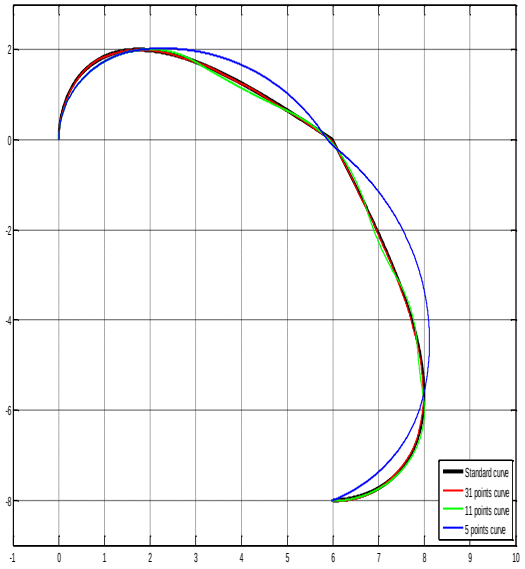


Fig. 6 - Simulatios of A to F trajectory using 0.2 as aid points

7. CONCLUSION

The followed methodology allowed successful simulation of curved trajectories that can be used in various scenarios, these behaved as expected, showing efficiency and accuracy during trajectory planning.

The number of points used depends of applications, because using less points results in trajectories less accurate but faster. In applications that need a more accurate trajectory is suitable use rotation in acute curves and a lot of control points. However, as mobile robotic don't need very accurate trajectories, with some exceptions, the use of less points could be interesting, since reduces processing and executing time and the results are satisfactory, as noted in complete path with 41 points (Fig.8).

When the trajectory planning calculates acute curves, the better path is approached by considering the aiding points because it has a better behavior since does not need to stop, and obtain faster response than others approaches.

The trajectory planning achieved allows the development and implementation of further toolbox blocks for calculating personalized paths for the mobile robot. It can also be used for developing embedded systems given Matlab ® environmental characteristics.

Future implementation of this planning trajectory in a robot is interesting to validate the trajectory planning in real world and use it in diverse applications, from industrial to home scenarios.

ACKNOWLEDGMENTS

The authors would like acknowledge the support of CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) and UNICAMP (Universidade Estadual de Campinas).

REFERENCES

[1] Kristensen, S. et al, "Interactive Learning of World Model Information for a Service Robot", Sensor Based Intelligent Robots Vol 1724/1999, pp.49-67, 1999.

[2] Roomba Homepage. Available in: <http://www.roomba. com.au>. Access in: mar.8, 2010.

[3] Automower's Husqvarna Homepage. Available in: <http://www.husqvarna.com/us/homeowner/products/robotic-mowers/husqvarna-robotic-mowers-for-homeowners>. Access in: mar.8, 2010.

[4] Information and Robot Technology at the University of Tokyo Press Release. Available in: <http://www.irt.i.u-tokyo.ac.jp/news/pressrelease081024_e.pdf>. Access in: mar.10, 2010.

[5] Fujita, M. and Kitano, H. "Development of an Autonomous Quadruped Robot for Robot Entertainment", Autonomous Robot, Vol. 5, No. 1, March 1998.

[6] Evans, J. "Helpmate: An Autonomous Mobile Robot Courier for Hospitals", Proceedings IROS, 1994

[7] Blaha,G. and Gerig, S.R. "Accurate Navigation of Industrial Mobile Robots",IEEE Position Location and Navigation Symposium, pp. 581-585, 1994.

[8] Jenkin et al., "ARK:Autonomousmobile robot for an industrial environment", In Proc. IEEE/RSJ IROS, Munich, 1994.

[9] Roth, H. and Schilling, K. "Navigation and Docking Maneuvers of Mobile Robots in Industrial Environments", IEEE, Conference of Industrial Electronics Society, Aachen, Germany, pp. 2458-2462, 1998.

[10] You, B. et al., "Development of a Home Service Robot 'ISSAC'", Proc. of the 1994 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Las Vegas, Nevada, pp. 2630—2635, 2003

[11] Dudek, G., Freedman, P. and Hadjres, S. "Using local information in a non-local way for mapping graph-like worlds," in Proc. of the 13th Int. Joint Conf. on Artificial Intelligence (IJCAI-93), pp. 1639–1645, 1993.

[12] Remolina E. and Kuipers, B., "Towards a general theory of topological maps," Artificial Intelligence, vol. 152, no. 1, pp. 47–104, 2004.

[13] Laboratory of Robotics of University of Heidelberg Homepage. Available in: <http://ornella.iwr.uni-

heidelberg.de/ROBOTICSLAB/ROBPROJECTS/COMPLETED/ASURO_LABY/D/>. Access in: mar.12, 2010.

[14] Institute of Robotics and Mechatronics at DLR Homepage. Available in: <http://www.dlr.de/rm/en/desktopdefault.aspx/tabid-3848/6062_read-9032/>. Access in: mar.12, 2010.

[15] Dudek, G. and Jenkin, M., "Computational Principles of Mobile Robotics", Cambridge University Press, 2000.

[16] MATLAB Central Homepage. Available in: <http://www.mathworks.com/matlabcentral/fileexchange/13332>. Access in: mar.12, 2010.